COMPUTER arts special

The in-depth guide for digital creatives

FLASH GAMES

100 PAGE GUIDE

Create and share playable games

Step-by-step tutorials

Over 50 pages on how to create Tetris, Asteroids and Space Invaders-style games

Techniques

- *Integrate audio into your games*
- *Lip-sync speech to mouth moves*
- *Design an intelligent preloader*
- *Create a game using video clips*

UK edition · Printed in the UK · Outside the UK and Eire £6

ISSUE 40 2002 £6

future publishing Media with passion

9 771465 112010 13

Our Data Centre. Your peace of mind.

Step 1:
Select Operating System

Cobalt RaQ

from £74.99 per month
The original "Web-in-a-box" solution.

Red Hat Linux

from £99.99 per month
Red Hat Linux is the industry standard for web hosting.

Microsoft Windows

from £199.99 per month
Microsoft Windows, the choice of small and large businesses alike.

Sun Solaris

from £199.99 per month
The market leader for reliability and performance.

Co-location

from £99.99 per month
Your servers safe in our Data Centre.

Step 2:
Select Specification

RaQ 128
128MB/20GB/**5 GB** Data Transfer

RaQ 256
256MB/20.4GB/**20 GB** Data Transfer NEW

RaQ 512
512MB/30GB/**30 GB** Data Transfer NEW

RaQ 550 NEW
256MB/40GB/**15 GB** Data Transfer
512MB/2x80GB/**25 GB** Data Transfer

Entry NEW
Celeron 1Ghz/128MB/40GB/**10 GB** Data Transfer

Professional NEW
PIII 1.13Ghz/256MB/60GB/**20 GB** Data Transfer

Ultra NEW
PIII 1.266Ghz/512MB/80GB/**30 GB** Data Transfer

Ultra II NEW
2xPIII 1.4Ghz/512MB/2x18GB SCSI Hard Disks running RAID 1/**35 GB** Data Transfer

Enterprise NEW
2xAMD Athlon 2Ghz/1GB/4x36GB SCSI Hard Disks running RAID 5/**55 GB** Data Transfer

Enterprise Xeon NEW
2xIntel® Xeon® 2Ghz/2GB/4x72GB SCSI Hard Disks running RAID 5/**80 GB** Data Transfer

Advance
2xUltra Web Servers/Ultra II Database & Ultra II Mail Server/**100 GB** Data Transfer

Corporate
3xUltra II Web Servers/Ultra Mail Server/ Enterprise Database Server/
250 GB Data Transfer

Corporate Advanced
4xEnterprise Web Servers/Xeon Database Server/Enterprise Mail Server/
400 GB Data Transfer

Sun Fire V100 128 NEW
500MHZ/128MB/40GB EIDE/
20 GB Data Transfer

Sun Fire V100 128 1GB NEW
500MHZ/1GB/2x40GB EIDE/
30 GB Data Transfer

Co-location
Your server safe and secure in our Data Centre. Choose from a range of data transfer (bandwidth) per month and benefit from 24/7/365 unlimited technical support.

Step 3:
Select Security

Firewall
only £100 per month

Protect your machine from unwanted visitors. Web based control panel allows you to add rules and deny undesirable visitors.

Backup
Daily - only £120 per month
Weekly - only £60 per month

Running databases, holding customer information or involved in e-commerce? If your answer is "yes" then order backup now and benefit from free restores if required!

Advanced Systems Monitoring
only £30 per month

The ultimate tool for knowing how your server is operating and running. Pre-empt critical failure and minimise server downtime with messaging via e-mail or SMS.

Plus Package
only £199.99 per month
(Save £600 per year)

Firewall, Daily Backup and Advanced Systems Monitoring bundled to save over £600 per year! The complete solution to your Web hosting security.

Step 4:
Relax with...

FREE AND UNLIMITED Technical Support

Unlike most of our competitors we do not charge for technical support or aftersales service. Available with every Dedicated Servers product, you can call or e-mail our support team on an unlimited basis, 24/7/365.

Service Level Agreement

Standard SLA:
Our Industry leading SLA is your guarantee of a totally reliable service. We'll pay you if it's not! Benefit from 99.9% connectivity, 30 minute reboot guarantee, 1:1 contention ratio on bandwidth and a price freeze-forever!

Advanced SLA:
Specially tailored for Enterprise level services and above, the Advanced SLA is perfect for mission critical sites. Backed by a money back guarantee, benefit from 99.999% connectivity, two hour hardware resolution, 30 minute reboot guarantee, 1:1 contention ratio on bandwidth and a price freeze-forever!

t:0870 333 0919 e:sales@dedicated-servers.co.uk

supporting

UK online for business

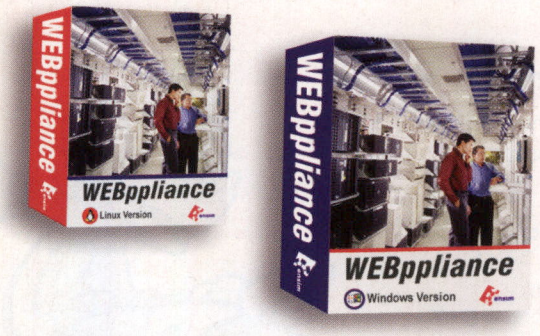
PART OF THE Host Europe GROUP

Product Focus

Ensim

ensim

FREE!



Free Ensim software pre-installed and ready to run on every Linux and Windows server!

This industry-leading, web hosting platform allows our resellers and customers to manage their hosting environment and administer their server with no technical ability. Ownership of the administrative functions and configuration of your own server will delight your customers, maximize performance and minimize costs. Ensim will transform your Dedicated Server into an easy-to-use, web hosting appliance that's ideal for web developers, designers and single site users alike.

For resellers:

- Set-up shared hosting accounts
- Control panels for your customers
- Integrated DNS support
- Full bandwidth management
- Customisable private label control panels

For single site users:

- Your own server (no sharing!)
- Easy back-up & restore
- Web-based e-mail
- Detailed on-line help & manual
- Real-time web-based control panels
- SSH Access

Free with Linux & Windows Servers

Step 5:

Contact Us

Order on-line or call Sales on:

t: **0870 333 0919**

e: **sales@dedicated-servers.co.uk**

w: **www.dedicated-servers.co.uk**

We constantly monitor the competition and know we offer the best value, feature-for-feature dedicated hosting in Europe. We are after all, the generic name associated with scaleable, secure hosting. We will beat any like for like quote...try us!

AMERICAN EXPRESS · MasterCard · VISA · SWITCH · DELTA · SOLO · DIRECT Debit

Benefit from...

Same Day Setup

Call us and place your order via credit card or direct debit before 4:00 p.m. (Monday-Friday) and we'll have your server up and running on the same day!

State of the art Data Centre

Our UK Data Centres are designed, owned and managed by us. Situated in London and Nottingham, they benefit from the latest security, backup, power and climate control features available.

Customer Charter

At Dedicated Servers, we are committed to providing you with the highest levels of customer service, covering all aspects of the customer relationship including: communications, accuracy, performance, privacy, security and complaints procedures.

Current Specials

Pre-Pay Bonus Pack

Our great new promotion gives you a huge bonus if you take out any dedicated server with a value of £2000 or more and pay for the year in advance. Each pack includes 5 books that will help you get the most out of your dedicated server as well as Host Europe merchandise. We will post out your bonus pack as soon as your server is setup.

FREE!

Advanced Systems Monitoring

For a strictly limited period, Dedicated Servers will enable Advanced Systems Monitoring on your dedicated server absolutely free, forever! Available with every server over £199 per month! Call now and don't miss out!

Our Data Centre. Your peace of mind.

Dedicated Servers

Prices exclude VAT @ 17.5% and setup where applicable. All trademarks acknowledged. E&OE

COMPUTER arts special
The in-depth guide for digital creatives

CONTENTS

AT A GLANCE

TUTORIALS
16 Game – Tetris-style puzzle game
26 Game – Space Rocks
34 Game – Space Invaders
44 Game – Animating characters
62 Game – Sound recording
66 Game – Video for games
70 Technique – Lip-synching
74 Technique – Preloaders
78 Technique – 3D characters

FEATURES
08 Top ten online Flash games
14 Essential ActionScript
20 Flash gaming book reviews

PROFILES
52 Kerb
86 Titoonic
92 Grey Digital

COMPETITION
MACROMEDIA
25 Win *Flash MX* software and books

REGULARS
07 CD contents
89 Back issues
90 Subscribe to CAS
96 Book reviews
98 Next issue

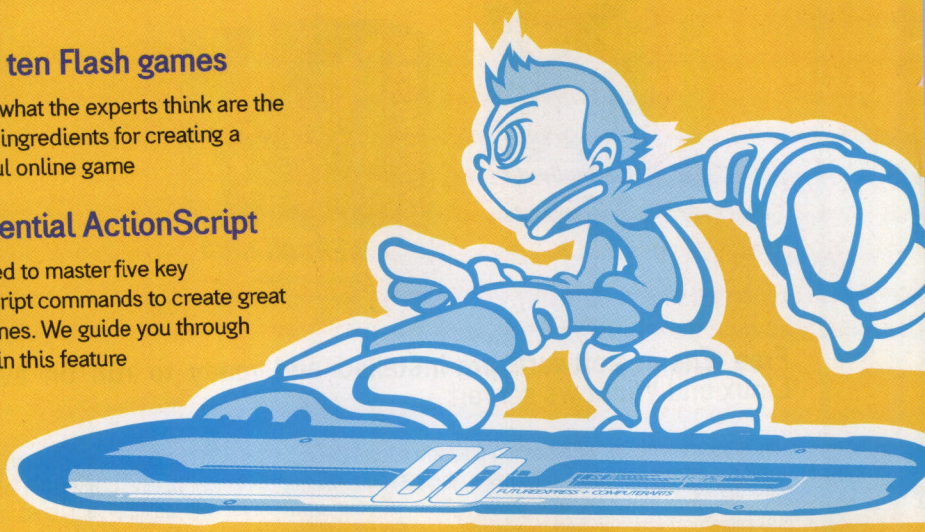
EXTENSIONS & GAMES
84 The best examples, all on our CD

FEATURES

Hints and tips from the best *Flash* game developers in the world

08 Top ten Flash games
Discover what the experts think are the essential ingredients for creating a successful online game

56 Essential ActionScript
You'll need to master five key ActionScript commands to create great *Flash* games. We guide you through them all in this feature



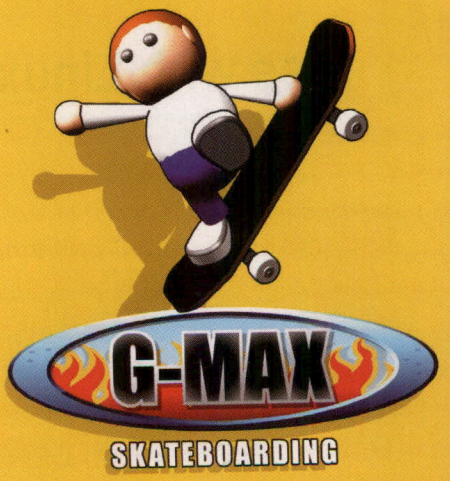
PROFILES

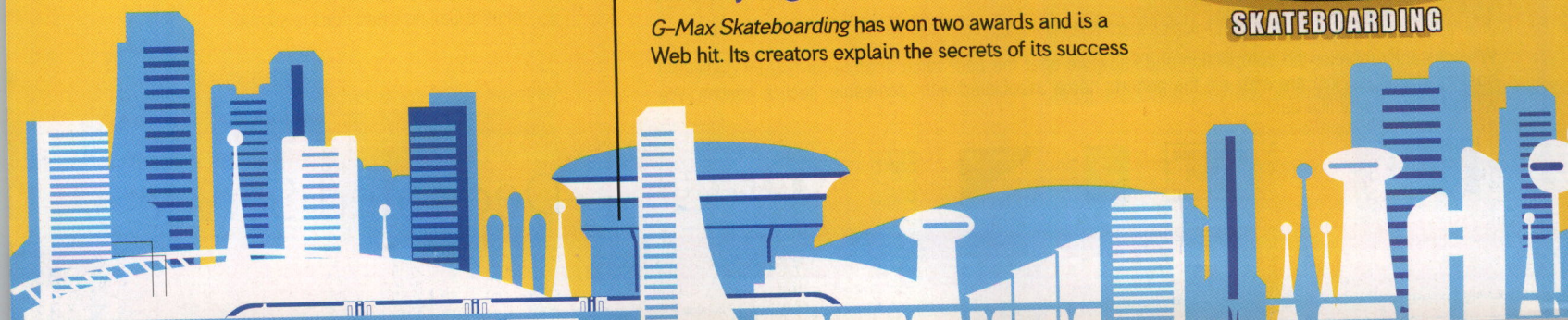
The lives and loves of real designers

52 Kerb
We find out why Brighton-based Kerb's *Flash* games have attracted the likes of Disney, BT and the BBC

86 Titoonic
One of the best-known games developers tells all about animation and inspiration

92 Grey Digital
G-Max Skateboarding has won two awards and is a Web hit. Its creators explain the secrets of its success

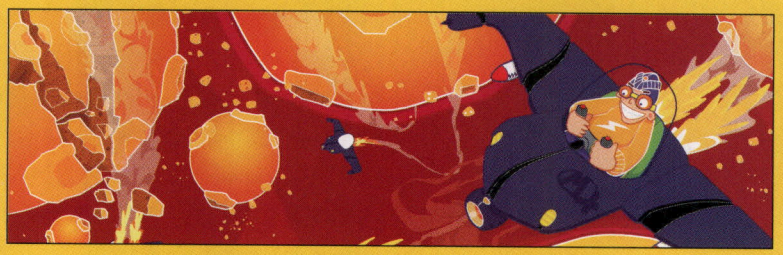




TUTORIALS

Follow our walkthroughs to create addictive online games

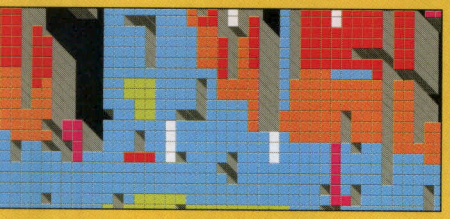
GAMES TUTORIALS



FLASH MX

26 Bring Space Rocks to life

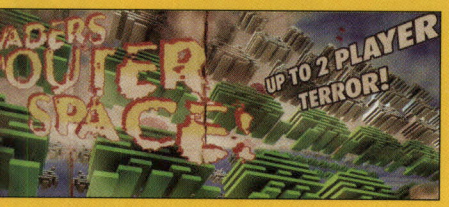
Asteroids is just as popular now as it was back in the 80s. We show you how to recreate its rock-blasting fun in your own version of this classic game



FLASH MX

16 Create your own Tetris-style game

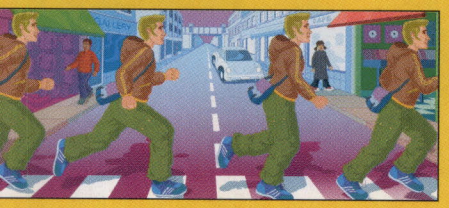
Make your own addictive, block-dropping puzzle game by following our comprehensive guide



FLASH MX

34 Recreate a Space Invaders game

Our step-by-step tutorial explains how you can produce a version of this legendary alien-invading game



FLASH MX

44 Animate your characters in Flash

We take a *FreeHand* character and show how you can bring it to life with an introduction to *Flash* animation

TECHNICAL SKILLS

FLASH MX

62 Make the most of game sounds

Here's how to record, prepare and incorporate audio as an intrinsic part of your games

FLASH MX

66 Use interactive video clips in games

Develop interactive video clips to produce a game that's suitable for virtually any machine or set up

FLASH AMP

70 Realistic lip-synching

Use *Flash Amp*'s unique features to get perfect lip-synching in your animations

FLASH MX

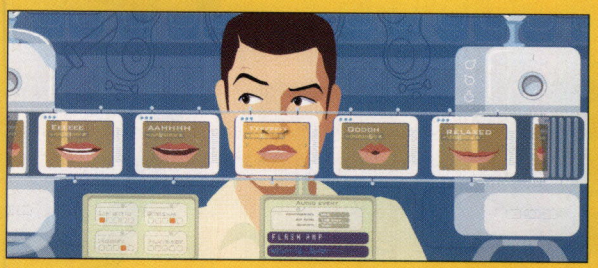
74 Preloader

Intelligent, well-designed preloaders are vital for larger *Flash* games. We show you how to create them

SWIFT 3D

78 Create 3D characters

Make a fearsome robot in our guide to producing your own animated 3D game characters



Get perfect lip-synching the fast and easy way with *Flash Amp*'s special features

Sleek performance from NEC/Mitsubishi

Incorporating the Mitsubishi SuperBright Diamondtron™ tube, the new Pro2070SB and Plus230SB 22inch CRT's deliver unmatched screen performance - optimum brightness, excellent contrast and outstanding colour uniformity.

The 22inch SuperBright Diamondtron™ range has been created for professional users in the CAD/CAM/CAE, Desktop Publishing and Web-Content Creation fields.

SuperBright Diamondtron

NaViSet™

sRGB

Quality is visible

08701 201160
www.nec-mitsubishi.com

NEC/MITSUBISHI
NEC-MITSUBISHI ELECTRONICS DISPLAY

SuperBright Diamondtron

COMPUTER arts special
The in-depth guide for digital creatives

SOFTWARE »

The essentials for *Flash* game design...

DUAL

Flash MX

We've got a full 30-day trial version of *Flash MX*, the best interactive animation software for flexible and powerful multimedia content, and the only choice for online games design

DUAL

Swift 3D v3

The demo of this great 3D software can be used to create fantastic game characters. We show you how to create a fully animated and textured robot in our tutorial starting on page 78

DUAL

FlashAmp

You can use the demo of this *Flash* add-on to lip-sync cartoon and game characters the quick and easy way. We show you how to get fast, realistic results in our tutorial from page 70

CD CONTENTS

Games

With our selection of the best games on the Web, there's bound to be something here to give you inspiration... and sore thumbs!
Go to page 85

Extensions

Get a head start in *Flash* game design with this collection of ten of the best Extensions around.
Go to page 84

£250 worth of sound samples

We've got an amazing selection of professional sound effects from the Sound Effects Library to help you make the most of audio in your games.

arts special
The in-depth guide for digital creatives

Subscriptions – page 41
Future Publishing Ltd, FREEPOST, BS4900, Somerton, Somerset, TA11 7B
[t] Customer hotline: 0870 444 8455
[f] 01458 274 378
[e] ca.subs@futurenet.co.uk

Editorial [t] 01225 442 244
Gillian Roach editor
[e] gillian.roach@futurenet.co.uk
Ashley Young acting production editor
[e] casproduction@futurenet.co.uk
Matthew Harvey art editor
[e] matthew.harvey@futurenet.co.uk
Matt Gallimore CD editor
[e] matt.gallimore@futurenet.co.uk
Joseph Russ CD editor
[e] joseph.russ@futurenet.co.uk

CD problems [t] 01225 822 743
Technical support ca.support@futurenet.co.uk

Contributors
Cover: FutureExpress

Illustrations: Thomas Brodahl, Brownjames, Peter Draper, Future Express, Stefan Gandl, Peter Greenwood, Rod Hunt, Magictorch, Ben Miners

Editorial: Michael Burns, Sprite Interactive, Kris Besley, CleverMedia, Mook, Niklas Alvaeus, Craig Grannell, Robert Carney, George Cairns, Chris Schmidt, Laura Sheppard, Jez Turner, Karl Hodge, Graeme Aymer, David Howell, Ashley Young

Printed at: TPL Printers (UK) Ltd.

Contact
Computer Arts Special, Future Publishing, 30 Monmouth Street
Bath BA1 2BW [t] 01225 442 244
[f] 01225 732 295 [i] ISDN: 01225 789 293
[w] www.computerarts.co.uk

Advertising [t] 01225 442 244
Rob Wittmaak Senior portfolio ad manager
[e] rob.wittmaak@futurenet.co.uk
Belinda Symington Key account manager
[e] belinda.symington@futurenet.co.uk
Bryony Watts senior account executive
[e] bryony.watts@futurenet.co.uk
Andrew Healy senior account executive
[e] andrew.healy@futurenet.co.uk
Lee Rossiter senior account executive
[e] lee.rossiter@futurenet.co.uk
George Lucas classified account executive
[e] george.lucas@futurenet.co.uk

Production
Lee Thomas production coordinator

Overseas licensing [t] 01225 442 244
[e] simon.wear@futurenet.co.uk 01225 442 244

Management
Sheena Pittaway publisher
John Weir publishing director
Colin Morrison chief managing director
Steve Jarratt group senior editor

Distribution
Seymour Distribution, 86 Newman St, London W1T 3EX [t] 0207 396 8000
Jamie Malley circulation manager
[e] jamie.malley@futurenet.co.uk

The Future Network
Roger Parry non-executive chairman
Greg Ingham chief executive
Colin Morrison ceo and managing director, UK
John Bowman group finance director

All rights reserved. All trademarks and copyrights in this issue are recognised, and are acknowledged where possible. If we have failed to credit your copyright then do please contact us – we'll happily correct any oversight. Any material submitted is accepted on the basis of a worldwide right to publish in printed or electronic form – be sure to clearly label all discs. All contents
© Future Publishing Ltd part of The Future Network plc which has offices in Bath, London, Milan, New York, Paris, San Francisco and Wroclaw.

Future Publishing Ltd is part of The Future Network plc.

The Future Network produces carefully targeted specialist magazines and Websites for groups of people who share a passion. We aim to satisfy their passion by creating titles that offer superb value for money. Today we publish more than 80 magazines and over 30 magazine Websites and networks from offices in five countries. The company also licenses 32 titles resulting in over 60 local editions in 23 countries.
The Future Network plc is a public company quoted on the London Stock Exchange (symbol: FNET) · Non-executive Chairman: Roger Parry · Chief Executive: Greg Ingham · Group Finance Director: John Bowman · Tel: +44 1225 442244 · www.thefuturenetwork.plc.uk · Bath London Los Angeles Milan New York Paris San Francisco Wroclaw

Computer Arts Specials is a member of the Audit Bureau of Circulations Jan-Dec 2001: **25,373**

Ten blinding Flash games

With *Flash* now a standard for online entertainment, there are more games created with it than ever before. But what makes a great online game? Is it originality, gameplay or a combination of all these factors and more? We jump into the fray and take a look at ten top games...

As Macromedia continues to develop *Flash* into an ever-more powerful rich media tool, there are greater opportunities for creating better and more diverse games. So what marks a game out to be great?

"First of all, it should load fairly fast," says Ferry Halim, creator of online games company Orisinal. "There are lots of people who are still on a 56k modem." Halim goes on to explain that the best games are simple, easy to learn, yet hard to master. Most important is the gameplay itself, and those games that have the ability to change dynamically as the player progresses are clear winners.

Glen Collins, Managing Director of Digital Outlook believes that a game has to be addictive. "There is always the opportunity to add in a multi-user aspect, which is hugely compelling," he explains.

The multi-user idea crops up repeatedly, with many gamers saying that this is the next stage of the *Flash* game, especially since Macromedia's release of the *Flash MX Communication Server*.

"Multi-player for me is what makes a game that I will keep and play for a long time — you just can't beat human interaction," says Jamie Madden, Creative Director of Australian developer BatteryHead. However, Madden says it'll be interesting to see how many people learn to harness the server since many *Flash* developers are designers, not coders.

Another key winning point is the subject of the game. Parodies of current affairs or those based on hot topics of the day make for very good viral games that spread across the Web like wildfire. Intelligent user interfaces are another plus. "Before releasing a game, you should let users test it and listen to their feedback," says Jobe Makar, Director of Electrotank. He also highlights the fact that the

BOOST!!

most popular games are often the best looking: "Graphics definitely make the first impression. Gorgeous graphics can help you get a lot of first-time players," he says, adding that the best games are also the ones with non-offensive sound effects and music, with the option to turn these off.

As well as the brainstorming sessions common to most online projects, developers often get their ideas from tried and tested games on other platforms.

"Most of our games have some sort of elements to them that have already been proven in some form or other," says Rene Boutin of Spore Productions, although he adds, "We stay away from what has obviously been done to death." Some developers steer clear of older gaming ideas altogether, though.

"There must be an original idea to build a real *Flash* game, not just creating on top of an old concept," says Nicolai Chauvet, Artistic Director of Studiotanuki. However, clients often want variations of games they've seen, and as budgets for *Flash* games are usually tiny, this is normally what they get.

Most designers also agree that it's hard to come up with purely original ideas. "In reality there are only a limited number of game types to work with, be it platform, maze, shooter and so on," says John Lyons, Interactive Director at Egovision. "What makes a game stand out is the twist that you put on it, the plot, characters and set."

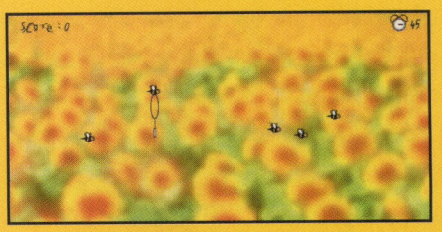
Contacts

StudioTanaki www.studiotanuki.com
Spore Productions www.sporeproductions.com
Wheel www.thewheelgroup.com
Kerb www.kerb.co.uk
Electrotank www.electrotank.com
Orisinal www.ferryhalim.com/orisinal
Hyperlaunch www.hyperlaunch.com
Digital Outlook www.digital-outlook.com
Edition Interactive www.editioninteractive.co.uk
Egovision www.egovision.com
BatteryHead www.batteryhead.com

Expertise provided by Michael Burns. You can contact Michael at [e] burns@dircon.co.uk

Bubble Bees

Company: **Orisinal**
Website: **www.orisinal.com/games/bubble.htm**
Designer: **Ferry Halim**
Development time: **Two weeks**



Bubble Bees is a popular *Flash* game in which players try to put flying bees into bubbles. The more bees you get, the better your score. There's also a time limit to the game.

"I was experimenting with the drag-and-drop feature in *Flash* when I got this idea for a game," explains designer Ferry Halim. "At first it was just a bubble-making game, but then I added the bees to make it more fun. I always try to make something different and new — it's not easy, but luckily, so far I've managed to produce some quite 'different' games."

Halim doesn't have specific stats for each game, but says that the Orisinal Website has around one million unique visitors each month.

What makes this a great online game?
"It has a different concept. It's fun. And it's the most popular game on Orisinal." ***Ferry Halim***

Barstar

Company: **Egovision**
Website: **http://polstarvodka.com/games/barstar**
Designer: **John Lyons (Art Director) and Anna Hague (Lead Illustrator)**
Development time: **Four months**

In this game you are a barman and the onus is on you to make a roaring success of a night out, keeping the punters, the owners and your own libido happy. The only way to do this is to make sure the punters get their favourite tipples without too much waiting.

"The game was for a luxury vodka brand with an established sexy image off-line," says John Lyons, Interactive Director at Egovision. "We devised the concept of a bar simulation game with a sexy twist, making the player rely upon a keen knowledge of cocktails and flirting to achieve a high score."

After concept and styling, the game engine was built to cross-reference internal databases of cocktails and a number of ingredients. The character AI and dialogue were then developed, and it was decided that the game would also form the basis of an enhanced data music CD with a *Flash* front-end, brand new cocktail recipes and a number of exclusive funky bar tracks.

What makes this a great online game?
"It was innovative and had a lot of depth. The cocktail list had over 100 entries, many of which were devised specifically for the game, and the visual style took the product into a new direction."
John Lyons

What the experts say

We asked several top developers to give us their ideas about the basic elements that every game should have...

"Multi-player; constant surprises; parody of current affairs; simplicity of interface; vivid scenarios; intelligent feedback on skill level; a leader board."
Jamie Madden, Creative Director at BatteryHead

"The primary element to any *Flash* game is a familiar control system. The control system should be something that the player has some knowledge of, or, if they haven't, it must be simple and quick to learn. It must also respond exactly how the player expects it to respond, and when they expect it to."
Mike Naman, Interactive Designer at Wheel

"'Email a friend' and player scoreboards. By combining great gameplay with multi-user functionality you've got the recipe for success with an online game."
Nick Kuh, Edition Interactive

"An increasing element of skill in the gameplay is important. When using a game in a wider viral marketing campaign, an element of competition is also important so people can forward it on to friends and see who's the best."
Don Jenkins, Marketing Director at Hyperlaunch New Media

"A concept, a goal and interactivity."
Ferry Halim, Orisinal

"Good design – consistent use of fonts, and colours and a careful execution of characters and background features."
Chris Clarke, Creative Director at Wheel

Stuntman Stunts

Company: **Hyperlaunch New Media**
Website: **www.hyperlaunch.com/stuntman/**
Designer: **James Humphrey (Creative Director) and Mark Birch (Production Director)**
Development time: **Four weeks**

Hyperlaunch New Media created a teaser game for the PlayStation 2 title *Stuntman*, featuring four different stunts designed to give the player an online feel for what the actual PS2 game is all about. The stunts are easy to get the hang of, but very hard to master.

"Ideally we look for elements of the title that we think would make a great game," says Marketing Director Don Jenkins. "A number of ideas are generated and storyboarded for the client. When an idea is agreed, assets are made and animation added. This asset is then handed over to technical in order to integrate the game into a site, link it to a high score table and add 'mail-a-mate' functions to it."

The game has been a great success. Since the launch of the campaign this year on 4 September, 60,000 visitors have so far clicked through to the game, with an average of 8,500 visitors a week since launch.

What makes this a great online game?
"The stunts are great fun and I think they really do give a great taster of the concept of an original PS2 title." ***Don Jenkins***

Pinpin

Company: **Studiotanuki**
Website: **www.studiotanuki.com/pinpin/**
Designer: **Meko**
Development time: **Two months**

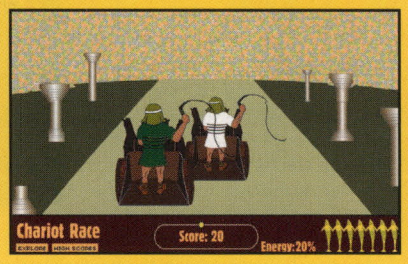
In *Pinpin*, you play a crazy rabbit who adds a rather personal brand of humour to a platform game. Users have to jump across stages, driving the rabbit around monsters using only the keyboard direction arrows. The Manga cartoon style is of very high quality, as is the amount of control the player has over the character of the flatulent rabbit.

"We wanted to create a lot of animations and possibility of moves for our character, so we built a script that allows animation for each instance," says Nicolai Chauvet, Artistic Director and Game Designer. "All our animations are made up entirely of *Flash* interpolations." This approach has been very successful and this wacky, high-quality game has received over one and half million hits in two months, with 200,000 pages served.

What makes this a great online game?
"Pinpin's responses, once the user understands how to command them." ***Nicolai Chauvet***

Chariot Race

Company: **Edition Interactive**
Website: **www.channel4.com/history/microsites/H/history/rome/index1.html**
Designer: **Nick Kuh**
Development time: **Four weeks**



Chariot Race is a 3D action game designed to boost traffic to Channel 4's History site. The player clicks around a 3D Roman Circus and moves into Play mode by zooming down into a packed arena. They start the game with 100 per cent energy, which is reduced each time they collide with debris or competitors, while points are gained for each avoided obstacle. The space bar activates the whip and players get bonus points if they whip their competitors as they overtake them. Edition's Nick Kuh designed the 3D interface and created the 'zooming' effect in *Swift 3D*, then exported it as an swf file to be brought into *Flash*.

"3D graphics tend to weigh down swf files so I had to make some sacrifices in export quality to keep the overall file size below 500KB," says Kuh. "The main challenge I met when programming was the collision detection script as this needs to activate only at the moment when the chariot racer appears to be passing the obstacles."

What makes this a great online game?
"The key to the popularity of this game is that it's fast paced and very difficult to achieve a good score. It is also a php/mySQL database-driven game with a high-score board." ***Nick Kuh***

Eelextrix

Company: **Wheel**
Website: **www.hulahoops.com/eelextrix**
Designer: **Catherine Gray**
Development time: **Three weeks**

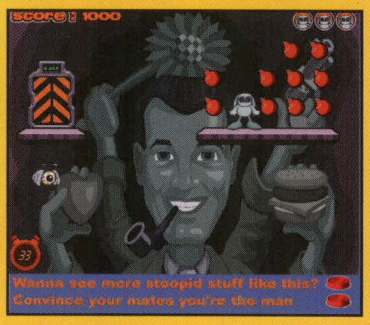
Based on the classic Scalextrix car-racing product, Wheel has developed an eel racing game for a Hula Hoops Shoks campaign. Players construct their own track from water pipes and race against an evil eel. A vicious plumber tries to make life even more complicated for the intrepid eel racers.

"We were briefed to develop online concepts that worked with the launch of Hula Hoops Shoks," says Chris Clarke, Creative Director at Wheel. "The advertising featured electric eels being released into the water system. We started thinking about eels struggling to escape from water pipes and came up with the idea of making the eels race each other. Classic compulsive games in the *Tetris* mould work well for older target audiences," Clarke continues. "For younger groups, an irreverent and slightly surreal design style is key."

What makes this a great online game?
"You can build your own track and race a friend. Also, there are lots of creative details that give the game a strong narrative flavour. " **Chris Clarke**

Panik

Company: **Kerb**
Website: **www.panikgames.com**
Designers: **Dave, Dan, Zeb and Dantoo at Kerb**
Development time: **One month**



This *Bombjack*-style game presents the player with three or four screens on each level where they have to collect all the bombs within a set time limit. Later levels have air ducts which need to be jumped on in order to access higher areas.

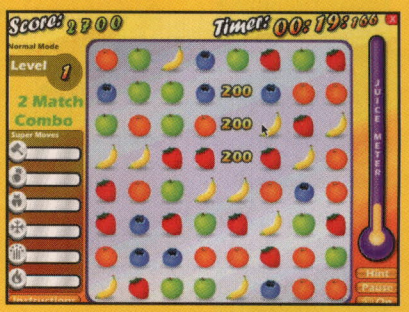
Panik's was conceived as part of a portfolio of games created in-house to license to third parties. After creating Panik the Rabbit's move controls, the team added the platforms, then the background and foreground graphics to the game. "We then fixed the baddies, then the bombs," says Jim McNiven, CEO of Kerb. "It was important to make sure that once the bombs were collected they didn't reappear when Panik went back to that screen."

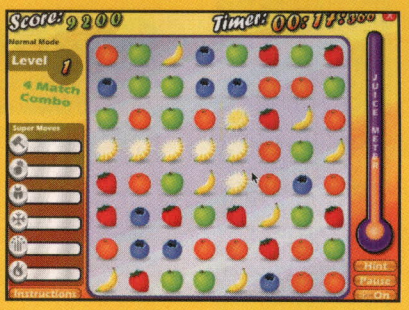
More tricky obstacles like the moving platforms and the conveyer belt with a saw blade were included as each new level was built. "With the bulk of the gameplay complete, we inserted the sound effects, sorted out the intro screen, preload and instructions at the start and a Game Over sequence at the end," says McNiven.

What makes this a great online game?
"We had it running on a PC in an arcade cabinet on our stand in Cannes (Milia) this year and people were playing it constantly. When people are on the stand asking you if it would be possible to create something similar for Websites, you get the feeling that it looks the part." ***Jim McNiven***

Fruit Smash

Company: **Electrotank**
Website: **www.electrotank.com**
Designer: **Robert Firebaugh (Creative Director)**
Development time: **Four weeks**





Fruit Smash is a grid-based fruit-matching puzzle game with a new and fun twist. Occasionally, instead of a new fruit, a different kind of icon will appear – a bomb, fire, geyser, ice, bug or mallet – representing one of six types of supermoves. A supermove icon acts like a wild card, so it can match with any two identical fruit.

"This game started with an experimental engine that handled everything but the supermoves," says Jobe Makar, Director of Electrotank. "We talked about the different directions that we could go with this game engine and decided to bring it to a higher level by adding in the idea of supermoves. When we settled on this idea we documented the entire game and then started to build it to completion." Development wasn't without problems though. Makar adds, "As with all games, along the way we decided on a few changes because of technical problems and playability problems."

What makes this a great online game?
"The addictive nature of searching more matches and applying supermoves. Supermoves are really what makes this game fun and unique." ***Jobe Makar***

"A good, playable game can be built out of blobs for graphics and beeps for sound effects. The quality of the graphics and effects then make a good game great. The most important thing is that when I press left the game object goes left and when I press fire the weapon is released."
Pete Scott, Chief Technical Officer and co-founder of Blue Beck

"Every *Flash* game should strive for simplicity and top quality gameplay. Sophisticated graphics and audio soundtracks are meaningless if the game is unplayable by all but the most patient (or bored) of people. Always consider that your audience is an entire planet of Internet users, including children, bored office workers and even retired men and women."
Rene Boutin, supreme leader at Spore Productions

"Beautiful graphics and solid ActionScript so that it feels as good as it looks. Half of the development time should be allocated to play testing, bug hunting and making the thing fun to play. There is a subtle line between challenging and irritating."
Jim McNiven, CEO at Kerb

"A good concept delivered well, easy to understand rules and controls and a considered interface are prerequisites. Great visuals make a good game great and shouldn't be underestimated, but gameplay is king."
John Lyons, Interactive Director at Egovision

"In one sentence: A good online game is one that loads fast, has nice graphics, fitting sounds, an intuitive UI, and can be played in a short time."
Jobe Makar, Director at Electrotank

Spore Cubes

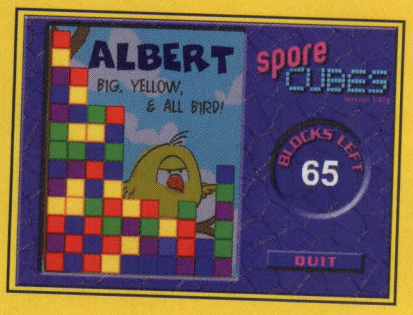
Company: **Spore Productions**
Website: **www.lilgames.com/cubes.shtml**
Designer: **Rene Boutin**
Development time: **Two weeks**

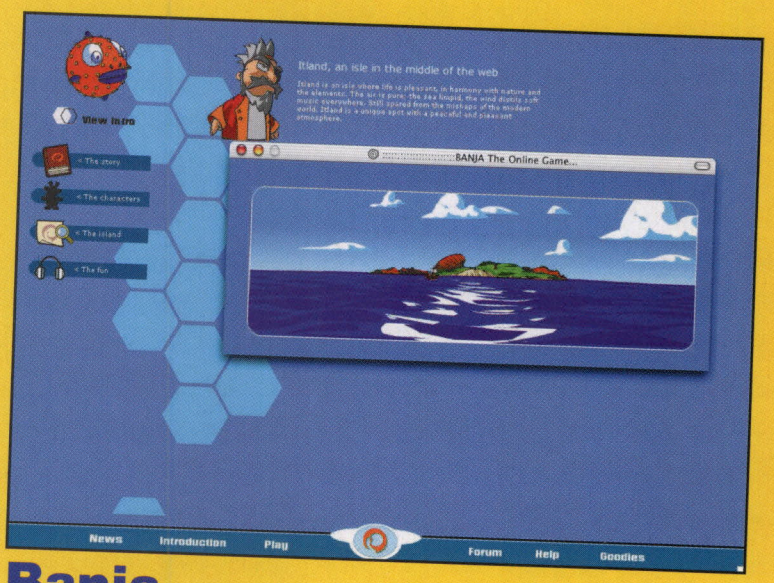
"A client wanted a couple of games where they could 'reveal' photographs gradually," says Rene Boutin of Spore Productions, explaining the origins of *Spore Cubes*. "They initially asked for a *Breakout/Arkanoid*-type clone, but that's an action game and is over-done." Boutin convinced the client to go with his new concept and it turned into a major hit.

"By now, the concept is easily recognisable as the 'click groups of same-coloured blocks to destroy them'," Boutin says. "The end-goal is to clear the entire screen of blocks." Boutin's inspiration was a shareware game which followed a similar concept. "I felt it was the type of game that could be simplified and would translate perfectly to an online format for the general mass market."

The game remains the most popular at Boutin's game site, LilGames.com. "Stats I've gathered show that a majority of the players are female and there is also an unexpected high number of seniors who love it," he reveals.

What makes this a great online game?
"Spore Cubes can be considered a veritable Flash game classic now. It was released in the summer of 1999 and shortly thereafter I started seeing clones of the concept appearing. Today, there doesn't seem to be a Web game developer or game site without some sort of variation of the concept. I humbly think the balance and simplicity of Spore Cubes keeps it as one of the more addictive of its genre."
Rene Boutin





Banja

Company: **TeamChman**
Website: **www.banja.com**
Designer: **Seb Kochman, Stephan Logier, Tony Derbome, Gauthier Malou and several others**
Development time: **Two years of initial production, then updated constantly since 1998**

Banja is the first community adventure game in 3D *Flash*, evolving in real time and exclusively on the Web. The goal is to explore ItLand as you attempt to help Banja keep the magical island pure and vibrant, discovering new games, places, music and surprises with every episode. Chat rooms and other integrated community tools, as well as arcade games and a media server, all add to the user's gaming experience.

Damien Giard of chmAN, *Banja*'s developer, says the team are all hardcore gamers from the Web design world who spent extensive amounts of time on LAN games, picking up tips on what was effective in a game, and what to leave out. In the creation of this graphics intensive world, special attention has been given to making the results as fluid for the player as for the viewer, who can watch the game as if it were an actual cartoon. *Banja* also integrates 3D-specific technologies, using *3DS Max* and *Flash* to ensure that there's a perfect render with both 3D depth and vector smoothness.

As for success, there are more than 200,000 registered players on the French servers, more than 150,000 on the Spanish/Latin-American servers and already more than 30,000 on the recently launched German servers.

What makes this a great online game?
"Banja is unique! There is nothing comparable on the Web or in any video game. From the adventure games, you collect objects, solve enigmas and discover new scenes with an emphasis on graphics and scenario. From the RPG games, it has dialogue and interaction with the characters, as well as a system of knowledge points. Mini game modules are also integrated in the story, with running, fishing, climbing, logging, motor biking, kiting and even a cooking game! The good spirit of Banja is also very important: positive and peaceful. Unlike most online games, the aim is not to conquer a territory or to exterminate the enemy, but to discover all the secrets of this mysterious isle, as well as living in a community, learning to control energy and preserving the environment." ***Damien Giard*** ca-s

LACIE

www.lacie.co.uk

External 'd2' Hard Drives & CD/DVD-RW's

- Stylish new design - arrange vertically, horizontally stacked or rackmount.
- Sturdy lightweight aluminium case for portability and durability.
- Choice of interfaces: Firewire (IEEE1394), USB 2.0 or U&I (USB 2.0 & Firewire)
- Simple hot-plug connection, instant drive recognition.
- Interface cables Firewire/USB included in bundle and price.

FireWire

HI-SPEED CERTIFIED USB

Discover LaCie's New Style

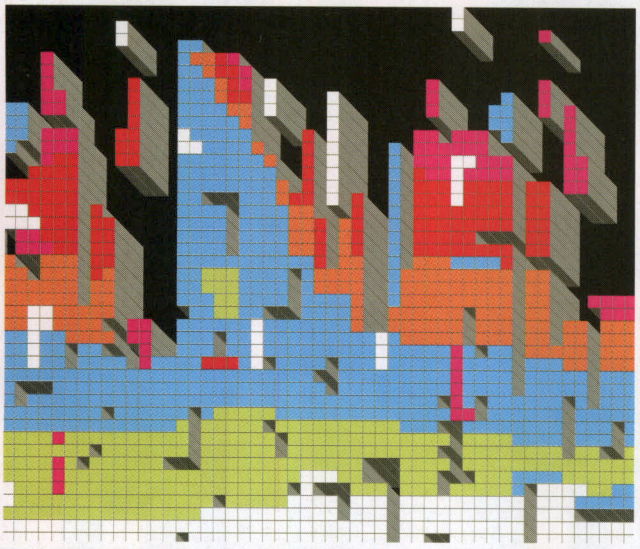
External Hard drives available in 60GB, 80GB, 120GB & 160GB. Fast 7200rpm suitable for DV video.
External archival drives available in 48x12x48x CD-RW, 4.7GB DVD-R/RW, 4.7GB DVD-R/RAM , AIT 1 Tape BackUp.

LaCie products are available from leading UK resellers.

visit: **www.lacie.co.uk**

Games introduction

Use Flash to recreate three of the most addictive games ever made, and learn how to animate characters for use in your own games...



16 Make your own arcade puzzler

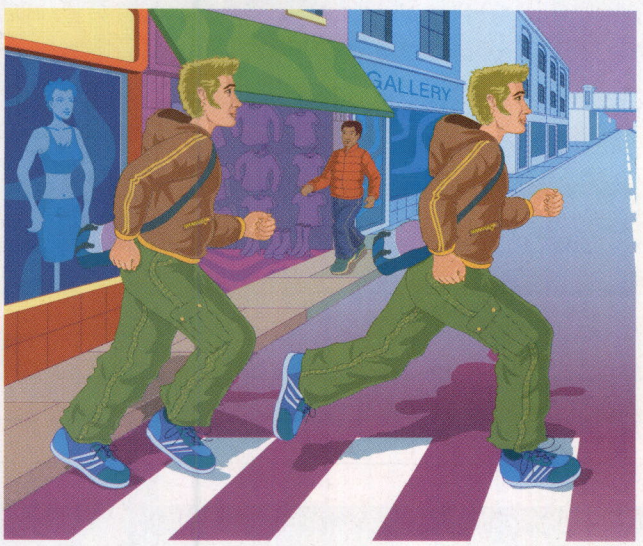
Tetris is one of the most well-known and popular games in the world. Here Sprite shows how you can create your own version of this block-dropping classic

26 Create an Asteroids-style game

Use *Flash MX* to produce *Space Rocks*, a version of the classic retro arcade shooter, with our comprehensive guide by Gary Rosenzweig of CleverMedia

34 Recreate Space Invaders

The concept behind this classic video game is simple, but making your own version will teach you the basic skills needed to design any game. Niklas Alvaeus explains...



44 Animate characters in Flash

Tom Evans and Tom Adams of Mook explain how to use textures, bitmaps and masks to bring a *FreeHand* drawing to life for use in your games

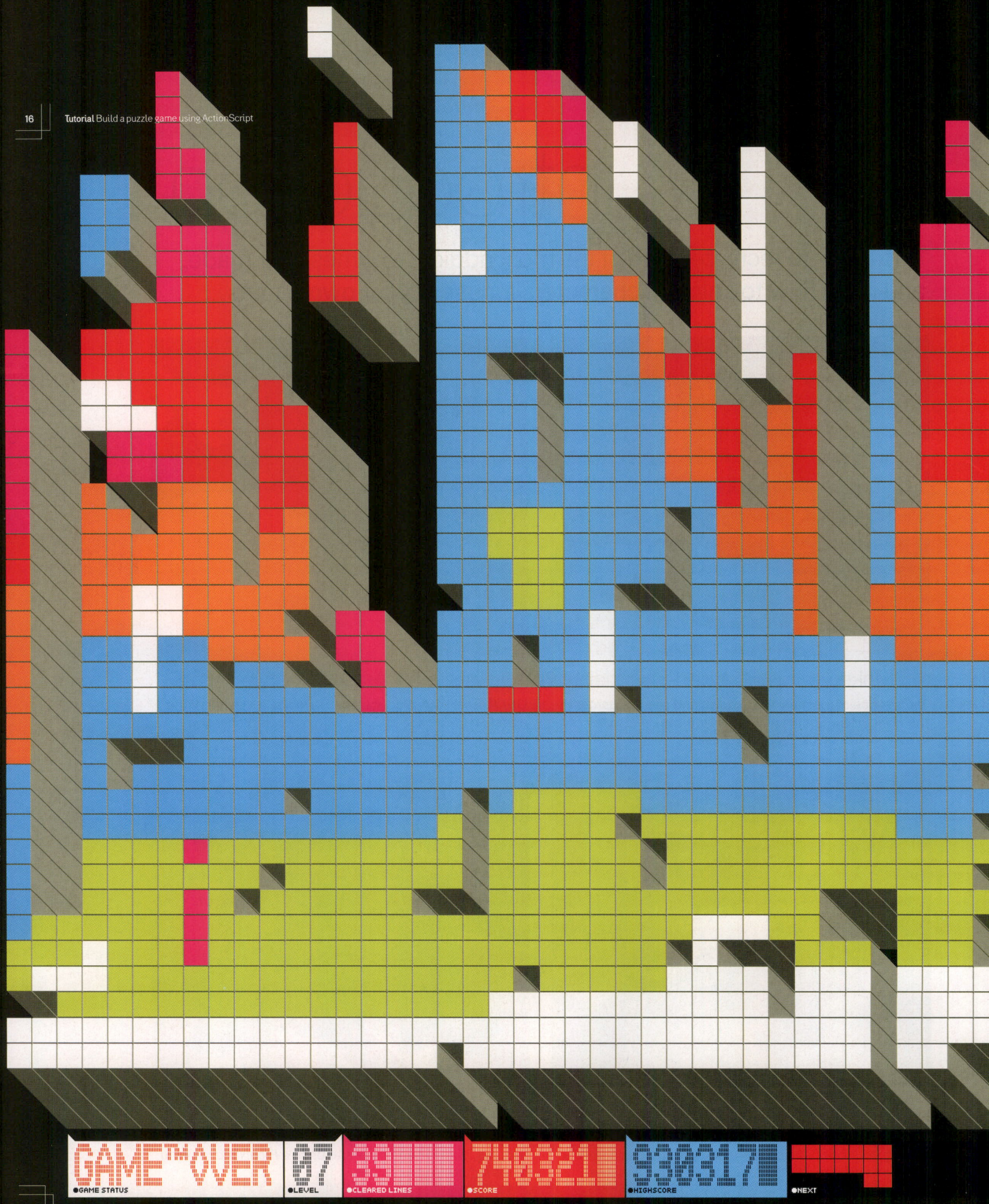


Illustration: Stefan@NeubauBerlin.com [w] www.NeubauBerlin.com

Build a Tetris-style Flash game using ActionScript

In this tutorial we'll show you how the object-oriented features of ActionScript can be used to create a versatile and modular game…

This might be the first time you have come across an object-oriented approach to ActionScript, so you need to understand the underlying concepts before you begin writing code. You need to understand what an object is, what a class is, how objects and classes are related and how they communicate by using messages. This approach is the same for all object-oriented languages and scripts.

What Is an object?

If you look around you now you'll see many examples of real-world objects, such as your dog, your computer, your phone or your book. These physical objects have two characteristics: they all have properties (variables) and behaviour (functions). For example, cats have different properties (name, colour, hunger) and behaviour (purring, meowing and licking).

In *Spriteris* we have a game board. This is the area on which the pieces are stored and which dictates the position and rotation of the active falling piece. The board has properties (rows, columns, pieces, lines cleared, level) and behaviour (move piece, rotate piece, shift piece, check board, create new piece and so on).

Software objects are modelled after real-world objects. A software object

This article and tutorial has been provided by Sprite, who are working on exciting projects in mobile marketing, VR and on the web. For more info, [e] alex@sprite.net [w] www.sprite.net.

Files on disc
All the files for the tutorial can be found on the CD. The *Flash* file is called spriteris.fla, and the code for the tutorial can be found in tutorials/gm_tetris

maintains its state in one or more properties (variables). A software object applies its behaviour as functions. The board object has an active piece, which is an instance of a piece object. The active piece is moved down the board until it lands on the bottom or another piece and then another piece is chosen randomly and created in order to start the process again.

In object-oriented programming, you might want to represent real-world cats as software objects in an animation program. You can also use software objects to model abstract concepts. For example, an event is a common object used in ActionScript to represent the action of a user pressing a mouse button or a key on the keyboard.

Everything that the software object knows (properties) and can do (behaviour) is expressed by the variables and the methods within that object. A software object that modelled your car would have variables that indicated the car's current state: its speed is 10mph, its engine speed is 2000rpm, and its current gear is fifth. These variables are formally known as instance variables because they contain the state for a particular car object, and in object-oriented terminology, a particular object is called an instance.

In addition to its variables, the software car would also have different methods to make it start, go forward, reverse, steer and brake. These methods are known as instance methods because they inspect or change the state of a particular car instance.

What is a class?

In the real world you often have many objects of the same kind. For example, your piece is just one of many pieces in the game. Using object-oriented terminology, we say that your piece object is an instance of the class of objects known as pieces. Pieces have properties (rotation, colour, structure) and behaviour (rotate, width, height, getData) in common. Each piece's properties are independent of and can be different from that of other pieces.

When building pieces, the programmer takes advantage of the fact that pieces share characteristics, building all the pieces from the same scheme. It wouldn't be very efficient to produce a new scheme for every individual piece created.

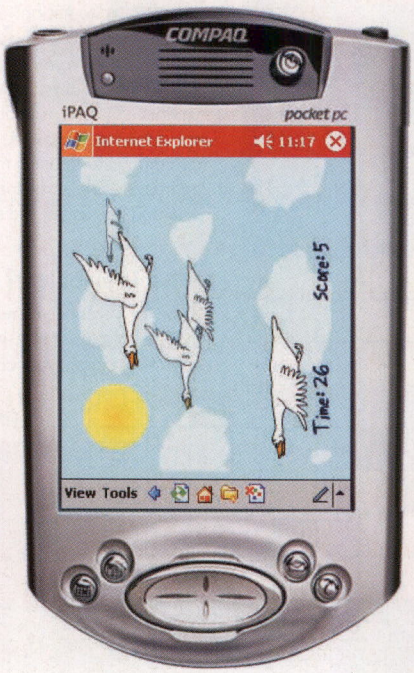
In object-oriented software, it's also possible to have many objects of the same kind that share characteristics. Like the 'piece' creator, you can take advantage of the fact that objects of the same kind are similar and you can create a scheme for those objects. A software scheme or blueprint for objects is called a class.

What is inheritance?

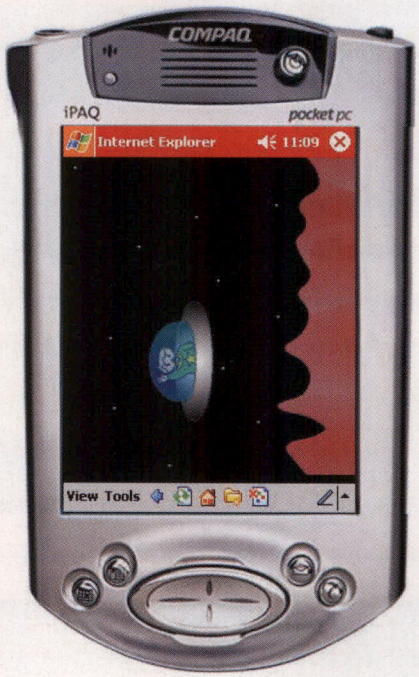
Generally speaking, objects are defined in terms of classes. You know a lot about an object by knowing its class. Even if you don't know what a Rover Mini is, if I told you it was a car, you would know that it had four wheels, two doors and a steering wheel. Object-oriented systems take this a step further and allow classes to be defined in terms of other classes. For example, BMW, Ford Escort and SAAB are all kinds of cars. In object-oriented terminology, BMW, Ford Escort and SAAB are all subclasses of the car class. Similarly, the car class is the superclass of BMW, Ford Escort and SAAB.

As you follow through the *Spriteris* game, we take you through a step-by-step introduction to designing and implementing a game using Object-Oriented ActionScript. If you're feeling adventurous you could even extend the game with your own pieces or create a multi-player version!

The same process of development also applies for Java development. The above illustration features Sprite's Tamogotchi pets running on mobile phones.



By developing in an object oriented environment it is very easy to develop for deployment on a number of different platforms.



The same methods used for developing content for a desktop PC can be applied to the Pocket PC, bearing in mind size and playback restrictions.

EASY TO BUY • EASY TO SET UP • EASY TO SEE

three gifts from the kings of domain names

Celebrate Christmas online with a cut price domain name from Easyspace. When you consider that we're one of the largest ICANN accredited registrars in the world and the UK's favourite domain name registrar, it's a wise choice. Offer ends 31st December. So log on while we're still bearing gifts.

log on today at: **www.easyspace.com**

Established in 1997 with over ½ million customers. Accredited ICANN registrar & nominet member. Prices exclude VAT.

easyspace
your perfect partner for the web

COMPLETE COMPUTER COVER

Future Publishing introduces a new deal for all computer owners:

Worldwide *All Risks* Computer Cover for theft or accidental damage to your computer equipment, including portable computers.

- No excess - full equipment value covered up to insured amount
- No additional security arrangements required
- Unlimited worldwide cover for portables, provided that the insured is resident in the UK
- Rapid claims response - authorisation can be within 24 hours
- This is an established scheme with one of the UK's leading insurers

For full details and to apply online: **www.computercover.biz**

Our readers are now able to have the best available protection for their computer equipment

ROYAL & SUNALLIANCE

www.computercover.biz

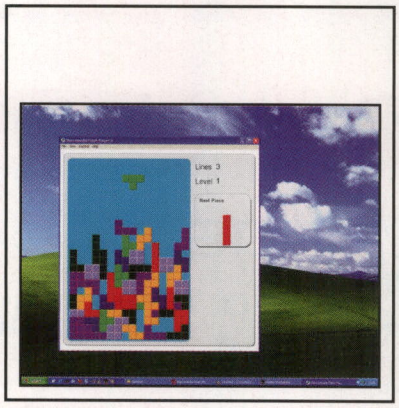
future publishing
Media with passion

Part 1: The game basics

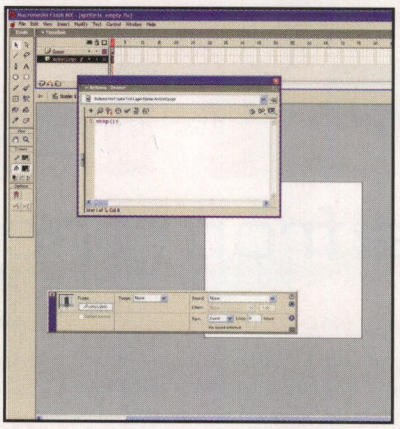
The first step is to create the building blocks of the game…

The history of Tetris

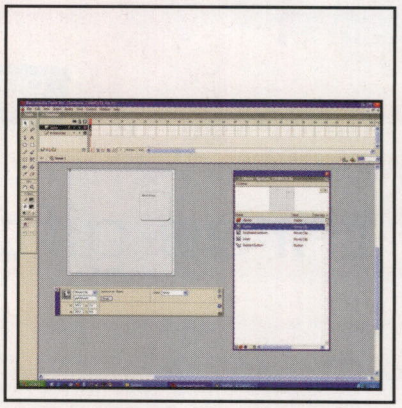
The game *Tetris* was first developed in Moscow in 1985. Later that year its creator, Alexey Pazhitnov, converted it for the IBM PC, and the game still remains a firm favourite among gamers worldwide. The arcade version of *Tetris*, released in 1987 through *Tetris*'s main developing and distributing company, Spectrum Holobyte, enjoyed extreme popularity and *Tetris* was included with all Nintendo Game Boy systems when the handheld was launched.



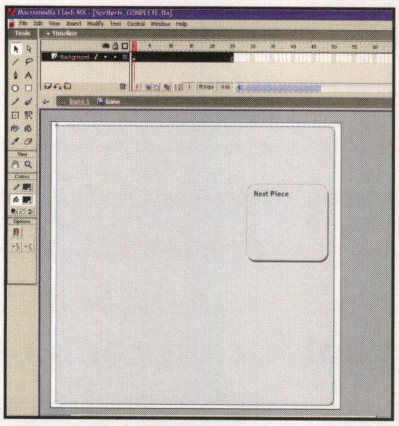
1 This tutorial will show you how to build a flexible *Tetris*-style arcade game using object-oriented ActionScript. Start by opening spriteris_empty.fla. This is an empty movie with the required library items.



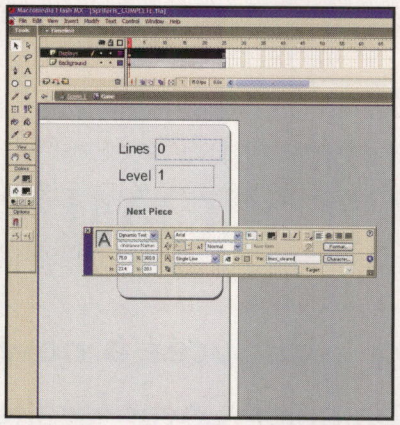
2 Create two layers named Game and ActionScript. On the ActionScript layer create a keyframe containing a single stop() action. The game will be stored in a separate movie clip.



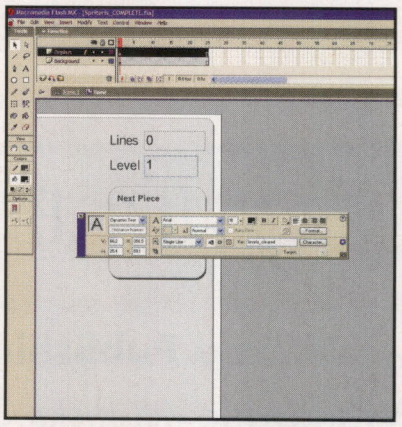
3 The next step is to create a movie clip to hold the game. By creating your game in a separate movie it's easy to add it to other *Flash* documents and scale to fit other movie sizes. Give the new movie the instance name 'gameboard'.



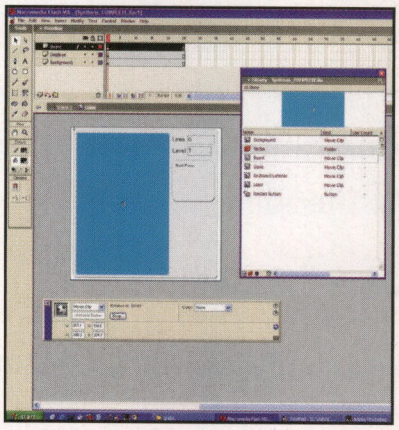
4 Open the gameboard movie and create a new layer named Background. Copy the movie clip 'Background' from the library onto the stage. This will be the static background for the game.



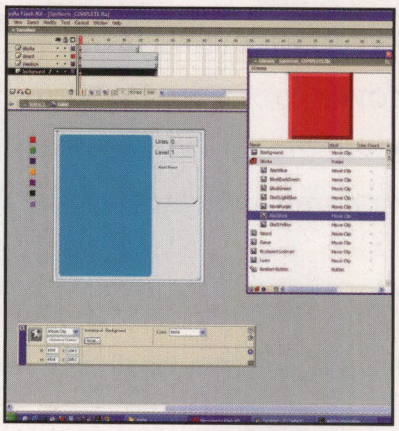
5 Create a layer named Displays and place two dynamic text fields on it, next to the Lines and Levels labels on the background. These will be used to display the status of the game.



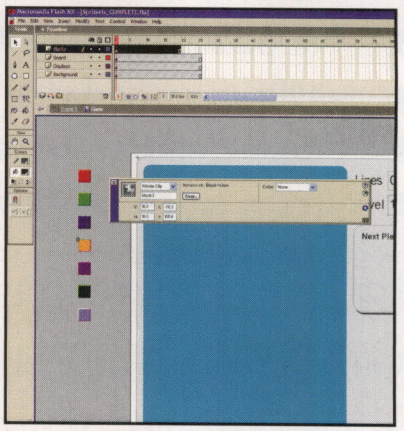
6 Give the Lines text field the 'Var' value 'lines_cleared' and the Level text field the value 'levels_cleared'. These variables will be used to update the text fields.



7 Create a new layer named Board and then place the Board movie clip from the library onto the first frame. This will be the background for the game board.



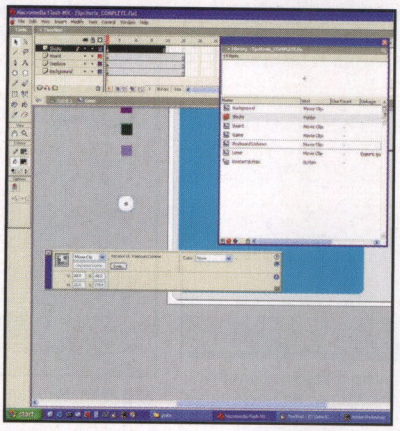
8 The pieces used in the game are built up from a number of simple blocks that are duplicated by ActionScript to display each piece. The building blocks are located in the Blocks folder in the Library.



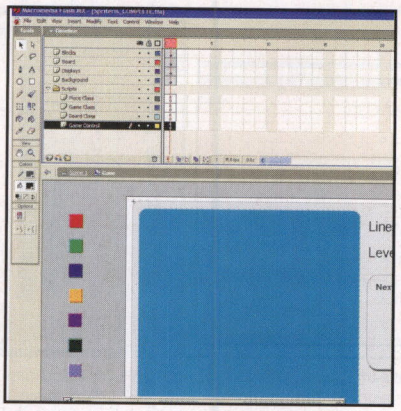
9 Create a new layer named Blocks and place an instance of each block from the library off to the left of the stage. Give each block the instance names block0, block1, block2… up to block6.

Part 2: Creating the game pieces

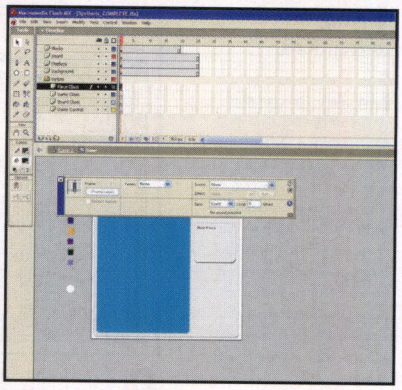
Next we take the plunge into ActionScript objects and create the Piece class...



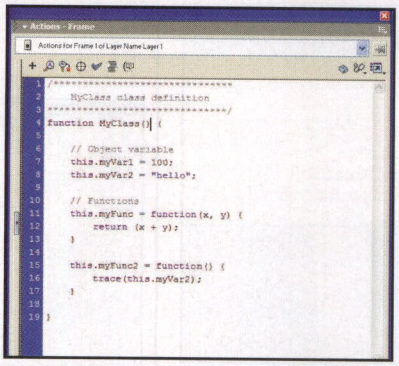
1 On the Blocks layer, create an instance of the 'Keyboard Listener' movie clip from the library. Later in the tutorial this movie clip will be used to detect keyboard events and perform game actions.



2 The whole game is based on object-oriented ActionScript, with ActionScript objects used to represent different elements of the game. The elements of this game are based on the Piece, Board and Game classes.

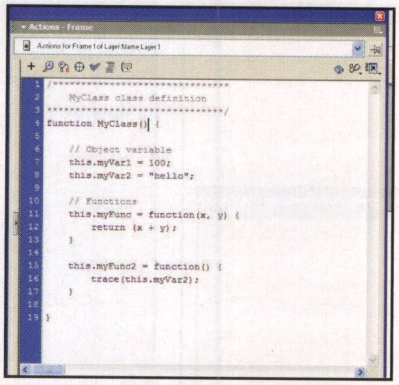


3 To help organise the code you should create four layers: Piece Class, Game Class, Board Class and Game Control. The first three layers will contain ActionScript used to define each object class while the game control layer is used to control the flow of the game.



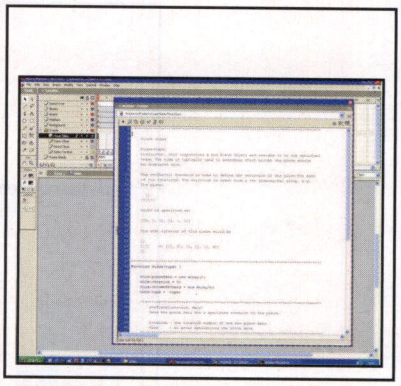
```
/*****************************
    MyClass class definition
*****************************/
function MyClass() {

    // Object variable
    this.myVar1 = 100;
    this.myVar2 = "hello";

    // Functions
    this.myFunc = function(x, y) {
        return (x + y);
    }

    this.myFunc2 = function() {
        trace(this.myVar2);
    }
}
```

4 A class is defined by creating a function with the same name as the class and placing variables and functions associated with that class within the function. For example: function MyClass() { ... }. To create a new object of the class you would use the 'new' keyword; for example: myObject = new MyClass();

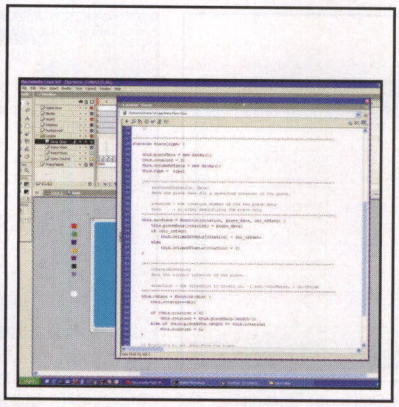


```
/*****************************
    MyClass class definition
*****************************/
function MyClass() {

    // Object variable
    this.myVar1 = 100;
    this.myVar2 = "hello";

    // Functions
    this.myFunc = function(x, y) {
        return (x + y);
    }

    this.myFunc2 = function() {
        trace(this.myVar2);
    }
}
```

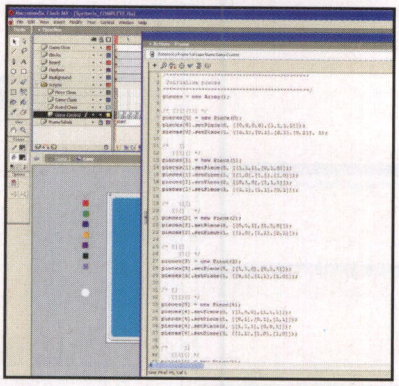
5 Within the class, variables can be defined by using the 'this' keyword; for example: this.type = 0;. Functions can also be defined within classes; for example: this.calculate = new function(x, y) { ... } This function could then be called from the new object; for example: myObject.calculate(1,2).



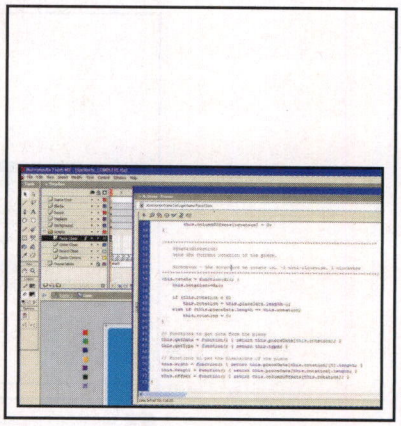
6 The first class to create is the Piece Class. A Piece object will be created for each different type of piece used in the game. In this case there are seven. Create a keyframe on frame 1 of the Piece Class layer and add the code from Code1.txt to the ActionScript window.



7 A Piece object contains the data for each rotation of that piece. The setPiece() function is used to set the piece structure for each rotation, the structure being specified using a two-dimensional array, similar to an image of the piece.



8 Create a keyframe on frame 1 of the Game Control layer and add the code from Code2.txt. This code is used to create each different piece used in the game and place them in an array. Refer to the comments in the Piece Class code for more details on the setPiece() function.



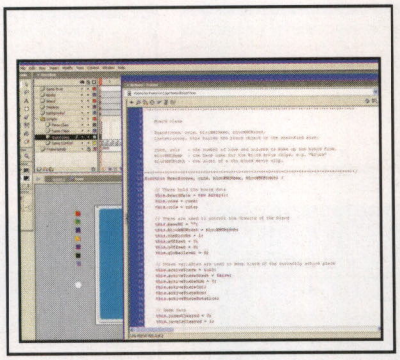
9 The Piece Class also offers functions to rotate the piece, get the array structure for the current rotation and retrieve other information about the piece, such as width, height and type.

Part 3: Creating the game board

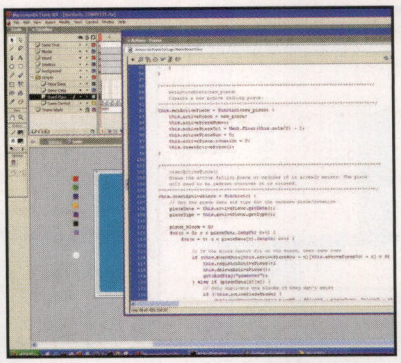
Now we'll look at creating the Board Class to hold the game board...

Naming conventions

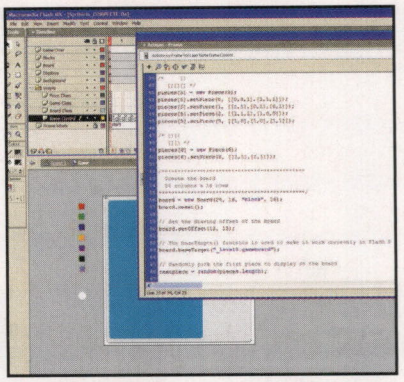
It's a good idea to learn the object-oriented programming naming conventions. A class always starts with a capital letter. When naming a function or a variable you need to start the name with a lower case letter, then if it is two words make sure the second word begins with an upper case letter. For example, movePiece(). You also need to give your variables meaningful names so that someone editing your code can come to grips with it easily.



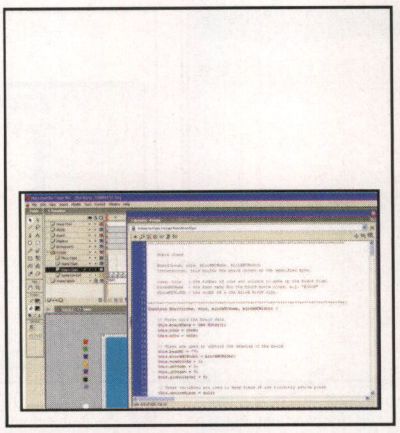
1 Now you have the game pieces, you need a board to place them on. For this, the Board Class is used. A Board object will manage the game board, control the active piece and check for complete lines and collisions. Create a keyframe on frame 1 of the Game Class layer and add the code from Code3.txt to the ActionScript window.



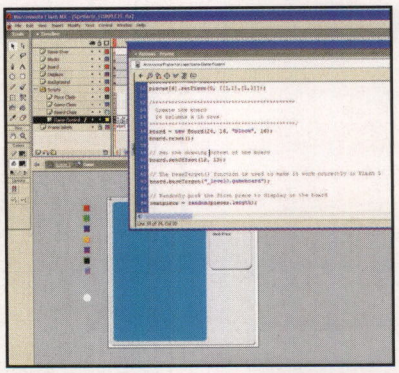
2 The Board class contains many functions that will be covered in this tutorial. When creating a new board, the number of rows and columns must be specified, along with the name of the block movie clip used to display the pieces, and the width of the block. The board creates an array to hold the status of the game board.



3 The reset() function of the board clears all the data from the board. The setOffset(x,y) function is used to align the display of the board from the top left corner of the main movie. On the first frame action of the Game Control layer, add the code from Code4.txt.

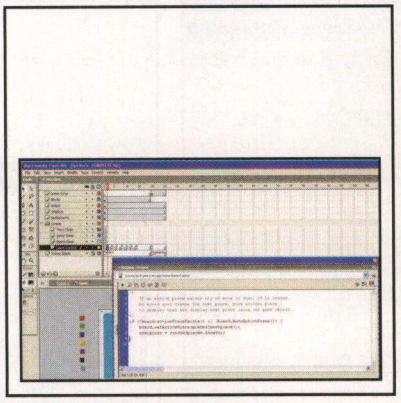


4 This code creates a new board of 24x16 blocks, assigns it to duplicate the "block#" MovieClips, resets it and sets the offset to display it at. This will form the basis of your game.

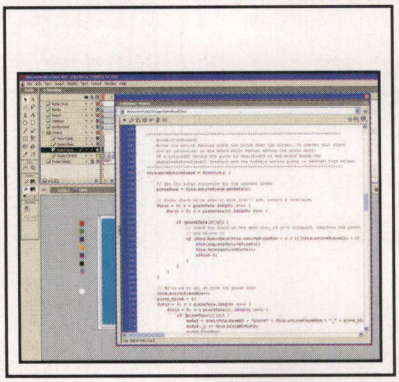


5 Finally add the code from Code5.txt to frame 1 on the Game Control layer. This will pick the first piece to display. The first frame of this layer is used to initialise the game, while the second frame performs the main game processes of moving the active piece, updating the board and creating new game pieces.

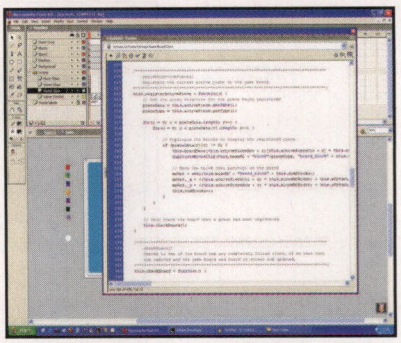
6 The Board object has an active piece, which is the piece being controlled by the player. To set a new active piece, the setActivePiece(newpiece) function is used to set it to the one specified. To check if an active piece already exists on the board, the activePieceExists() function is used.



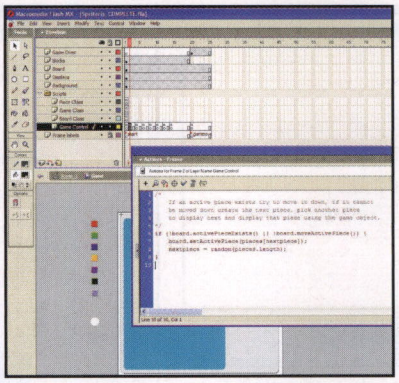
7 Create a new keyframe on frame 2 of the Game Control layer and add the code from Code6.txt to the ActionScript window. This code checks whether an active piece exists on the board, and if so, whether it can be moved down.



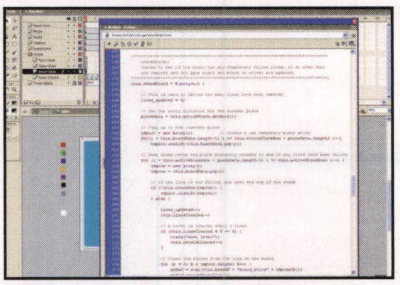
8 The board.moveActivePiece() function is responsible for moving the active piece down the board. It first checks for collisions with existing blocks before updating the display. If a collision occurs, the active piece is registered on the board and then deleted so another one can be created.



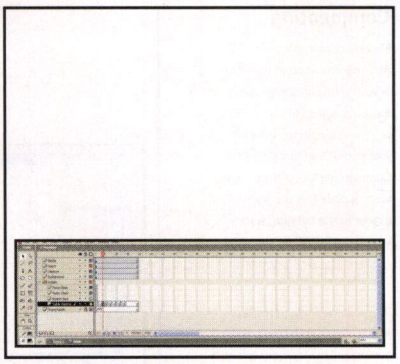
9 The Board class contains a two-dimensional array to represent the game board, where each element of the array corresponds to a block on the board. The registerActivePiece() function is responsible for registering the active piece in the array when the piece has landed. After it has been registered, the active piece is deleted using the deleteActivePiece() function.



10 If a new active piece is needed, the board.setActivePiece() function is called with the next piece to create. After it's created, another piece is randomly chosen to be the next piece.



11 At the end of the moveActivePiece() function, the checkBoard() function is called. This function is used to remove any filled rows from the board array and to update the game board on screen. Every time a row is cleared, the linesCleared variable of the board object is incremented. Each time five lines are cleared the levelsCleared variable is incremented.



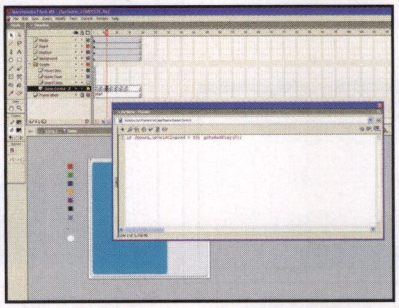
12 The levelsCleared variable is used to control the speed of the game, so the higher the level the faster the pieces fall. Create keyframes at frames 4, 6, 8, 10, 12 and 14 on the Game Control layer.

Part 4: Adding game and user controls

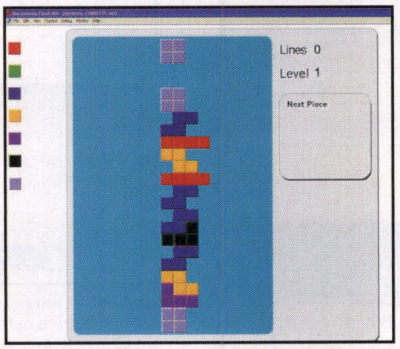
Adding keyboard control for the user and making the game more challenging…

Using layers

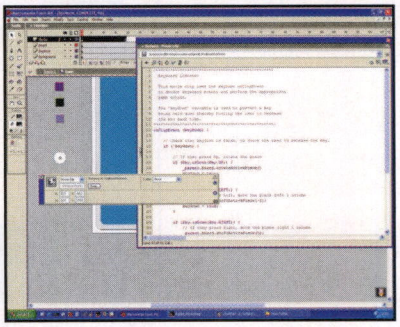
When building a movie of this complexity, you'll find it easier if you create each ActionScript class on its own layer. This helps to break up the structure of your movie, meaning you won't have to scroll through hundreds of lines of code to find the class you want. *Flash MX*'s new function that allows you to organise your layers into folders also proves useful, keeping your movie as uncluttered as possible.



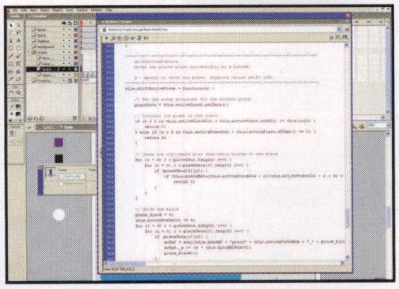
1 Open Code7.txt and follow the comments in the code to paste it into the appropriate frames on the *Flash* file. Each keyframe checks if the levelsCleared variable of the board object is above a certain level. If so, it'll immediately repeat frame 2 to drop the active piece down one step. The higher the level, the sooner it repeats.



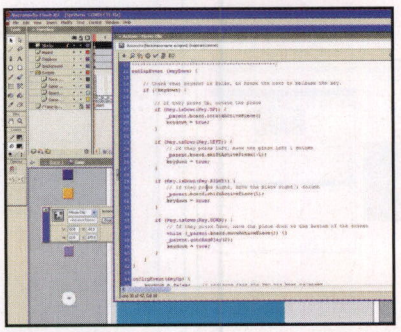
2 By testing the movie at this point you'll see that the pieces fall into a column on the board. The next step is to add some control for the player. The active piece will be controlled by the keyboard using the cursor keys.



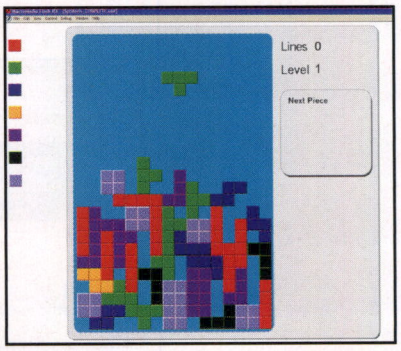
3 Select the Keyboard Listener movie clip on the Blocks layer. Open the Object Actions and paste into it the contents of Code8.txt. The code uses an onClipEvent(keyDown) { … } block to detect when a key is pressed, and Key.isDown() checks to find out what key was pressed.



4 When the left or right cursor key is pressed, the function board.shiftActivePiece(dir) is called. This function moves the active piece on the board in the direction specified, either -1 for left or 1 for right. Pressing the up cursor key calls the board.rotateActivePiece() function. This function rotates the active piece to its next position and updates the display.



5 The down cursor key is used to drop the active piece until it lands. It does this by repeating the board.moveActivePiece() function until it returns to 0, meaning that it has landed. When it has landed, the game is restarted at frame 2 to generate the next active piece.



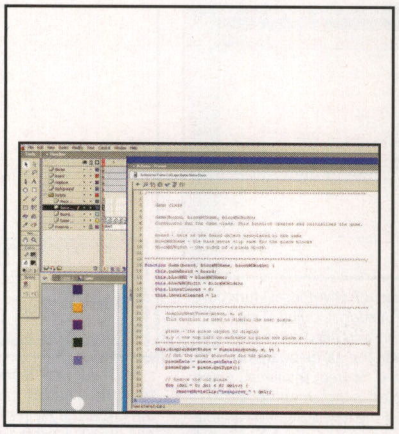
6 By testing the game at this point you'll see that it's getting more playable. The next step is to provide a display of the next piece and update the display of the number of lines and levels cleared. A Game object is used for this.

Part 5: Finishing touches

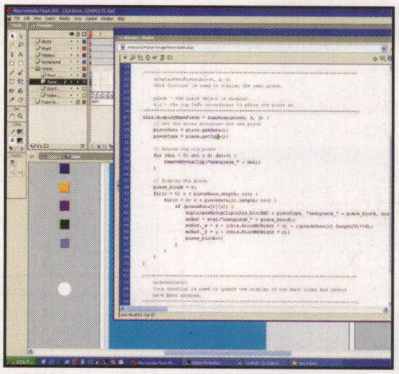
Adding the essential features to create the complete game…

Commenting

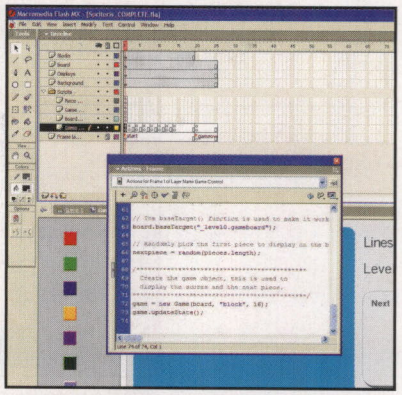
Make sure you comment all your ActionScript so other members of your team can understand your movie. You need to use '//' tags or '/*' ...'*/' blocks to comment your code. Don't forget that the Commenting tool is a great way to store code that you want to use in your movie but don't want turned on at the time you write it.



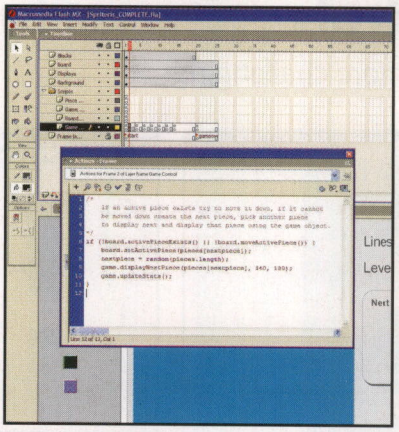
1 Create a keyframe on frame 1 of the Game Class layer and paste the code from Code9.txt into the ActionScript window.



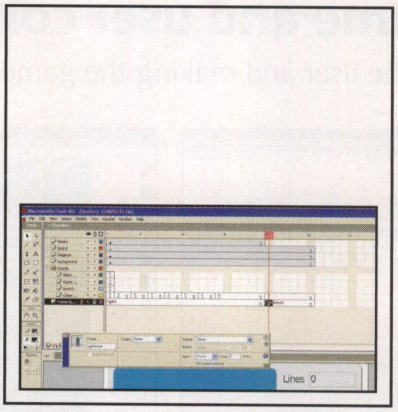
2 The Game Class provides two functions. The first is displayNextPiece(piece,x,y) and this is used to update the display of the next piece. It accepts three arguments: the first is the Piece object to display, and x, y are the top left co-ordinates to display the piece at.



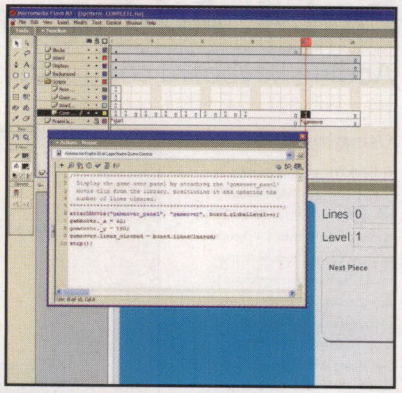
3 The second function, updateStats(), updates the Lines and Levels text fields to display the status of the game. To create a new Game object, add the code from Code10.txt to the end of the ActionScript on frame 1 of the Game Control layer.



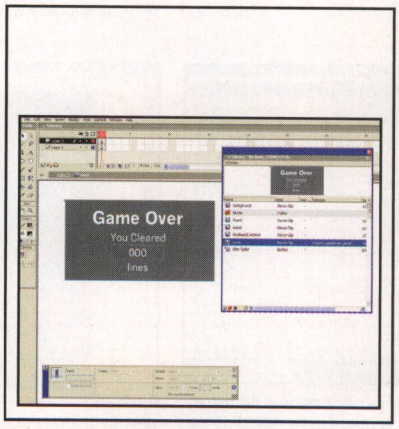
4 The Game Class functions need to be called on the second frame, whenever a new piece is created. Replace the code on frame 2 of the Game Control layer with the code in Code11.txt.



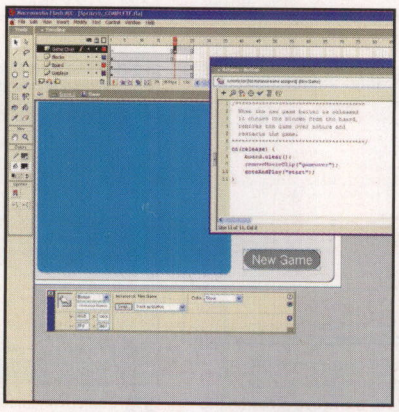
5 All that now remains is to control when the player runs out of board and it's Game Over. Create a new layer named Frame Labels, create a keyframe on frames 1 and 20 and give them the frame labels 'start' and 'gameover' respectively.



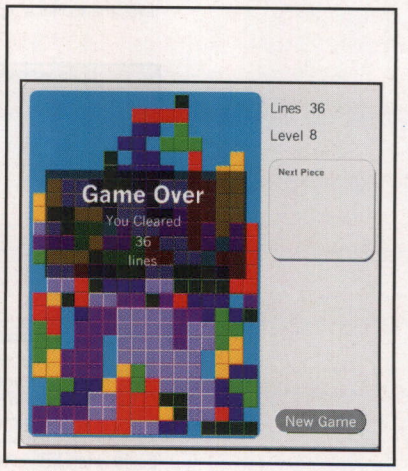
6 On frame 20 of the Game Control layer, add the code from Code12.txt. This code stops the game and displays the Game Over message. It does this by attaching a movie clip from the library with the linkage name gameover_panel. The message is then positioned over the board.



7 The message is attached dynamically to make it appear on top of the existing blocks on screen, since all graphics on the timeline will appear beneath duplicated movie clips.



8 Create a new layer named Game Over and a keyframe on frame 20. On this keyframe, place an instance of the New Game button from the library and then add the ActionScript from Code13.txt to it.



9 Now simply compile and play the game! Open spriteris.fla to see a complete version. ca-s

Win Flash MX, Flash Communication Server MX and two official books

Your chance to win this great package of *Flash* software and official books from Macromedia

We've teamed up with Macromedia to offer two of you the chance to win a fantastic bundle which includes a copy of *Flash MX* and *Flash Communication Server MX*, plus two great new books from Macromedia Press, worth over £70, to help you get the best out of *Flash* MX.

Flash MX

You'll recognise *Flash MX* as being quite simply the best interactive animation tool available, delivering fast and powerful video and multimedia content.

Flash Communication Server MX

This server software unites communications and applications for those who demand streaming media, real-time collaboration and rich-media in a powerful and open online environment. With Video and Audio Conferencing, Whiteboards and much more, it's the only choice for the widest possible online audience.

Macromedia Flash MX ActionScripting: ActionScripting from the Source **£33.99, ISBN 0201770229**

Derek Franklin and Jobe Makar teach ActionScript with a hands-on approach in this book, translating real-life activities into scripts and showing methodologies and techniques for building almost 30 ActionScript projects, from sample games and Websites to wireless applications. The book comes with a companion CD including all of the project files and images needed for the written tutorials.

Macromedia Flash MX Game Design Demystified **£37.99, ISBN 0201770210**

With simple explanations of basic logic games before moving on to multi-player games, 3D graphics, sound effects, licensing and more, this resource contains all you need to get started in games design using *Flash MX*. Its comprehensive coverage of sophisticated techniques is simply presented and the book comes with a CD including the source files for the games and tutorials that are covered. See page 97 for our review.

Question

What score did we award *Macromedia Flash MX Game Design Demystified* in our book review this issue?

To enter

The easiest way to enter is via our Website at **[w]** www.computerarts.co.uk/competitions. Or you can click the Competition tag on the Contents screen of our cover CD, then enter Macromedia as the competition name, and fill in the rest of the form. Click Submit Entry. Alternatively, send your answer on a postcard, along with your address and a contact number to: Macromedia Competition, *Computer Arts Special*, 30 Monmouth Street, Bath, BA1 2BW.

The rules

The competition closing date is 23 January 2003. Employees of Macromedia or Future Publishing, or any of their agents or families, may not enter. Multiple entries will not be accepted. The editor's decision is final. There are no cash alternatives. No other correspondence will be entered into. If you are entering by post and do not wish any of the companies involved in this competition to contact you with further offers, indicate this on your entry. We won't pass details on to third parties.

Previous winners – issue 38

Illustrator and *Streamline* winner – Tim Hayne from Bridgewater. Runners up – B. Entwistle from Salford, Sylvia Copleton from Glasgow, J. R. Osborne from Eastbourne, Irina Smith from Southampton.
CorelDraw 11 – J. McKeon from Washington, Ron Cottam from Alfreton, Joanne Van Campen from Tunbridge Wells.
FreeHand and *Illustrator* software training – John Muir from Glasgow, Peter Smart from Erith, Graham Pattrick from Beccles, G.O. Parry from Holyhead, Andrew Adams from Camerton, Chris Killingback from Loughborough. **ca·s**

For further information on all Macromedia products, visit **[w]** www.macromedia.com.

07
SHARK bite
GAS
COLA

Illustration: Brownjames [t] 0121 350 8561
[e] jimmyjames@btinternet.com [w] www.brownjames.co.uk

Recreate a classic arcade shooter

Use *Flash MX* to make an arcade space combat game featuring floating rocks and a spaceship built to destroy them...

With *Flash MX's* improved speed and new scripting features, it's now easier than ever to create arcade games for your Website. *Space Rocks* is a classic, and was one of a genre of arcade games that dominated in the early '80s and still manages to entertain a large number of players today.

The game features a small ship placed in the middle of a large field of floating rocks. The rocks whizz around the screen at various speeds – they wrap from one side of the screen to the other so they're never very far away.

A lone ship stands against these rocks. The player controls the ship's rotation and can thrust the vessel forward to avoid the rocks. The ship can also fire an endless amount of bullets to combat the rocks. The larger rocks break into smaller pieces, and eventually disappear completely.

To make a game like *Space Rocks*, you need *Flash MX*, the latest version of Macromedia's popular Web development tool. You also need to be familiar with the application because you'll be using some of its new scripting features during this tutorial.

Expertise by Gary Rosenzweig. Gary is the founder and Chief Engineer at CleverMedia **[w]** http://clevermedia.com/, a company that specialises in Flash and Shockwave games. His latest book is called *Flash MX ActionScript for Fun and Games*. You can find out about Gary's books and contact him through **[w]** http://garyrosenzweig.com/.

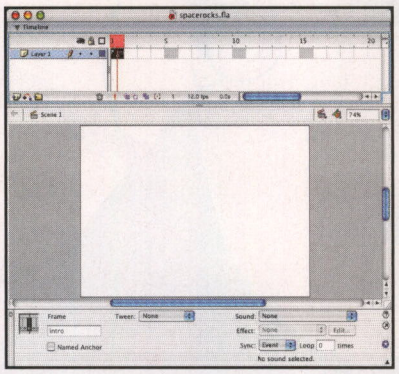
Files on disc
The files needed to support this tutorial can be found in the Tutorial\gm_rocks folder on the cover CD.

Part 1: Create the games arena

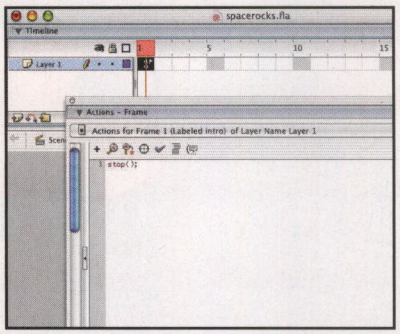
The first thing you need to do is create your game intro screen…

Starry skies

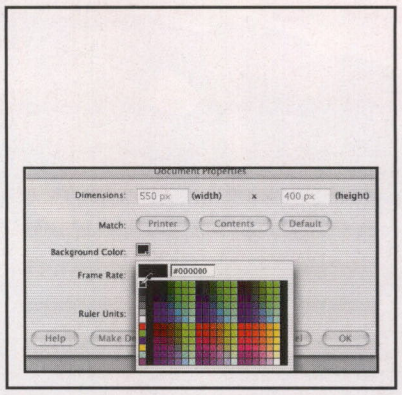
If you want to create a more interesting background than the plain black one used here, you can make a new layer in the main timeline and fill it with little dots to represent stars. Make sure these dots are very small, so they won't be confused with the rocks.



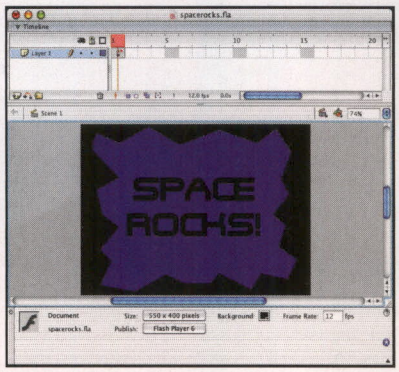
1 Open *Flash MX*. Select the first frame in the main timeline and use the Property Inspector to name the frame 'intro'. Save the movie as spacerocks.fla and remember to save after each step.



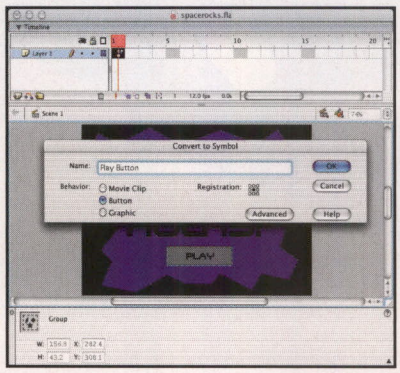
2 Press F9 to bring up the script panel while the first frame is selected. Enter the simple script stop(); to get the movie to pause on this frame when it starts. Without this script, the movie would play through all the frames we are about to create as if it were an animation, rather than a game.



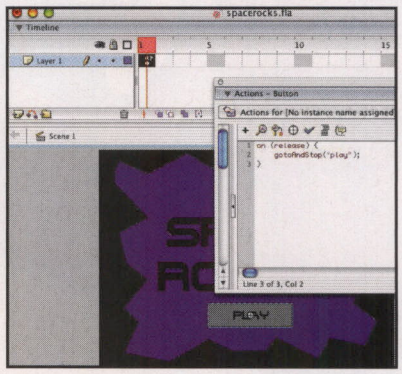
3 Open the movie Properties dialog by choosing Modify>Document. Click on the Background Colour chip and change it to black.



4 By using text and graphics of your own choice, draw on this first frame to create a title graphic and introduction. Later, you can add instructions and other content to this frame.



5 Create a button graphic on the stage. Select it and use F8 to convert it to a button. Name it anything you like. Make sure you select Button in the Convert to Symbol dialog box.



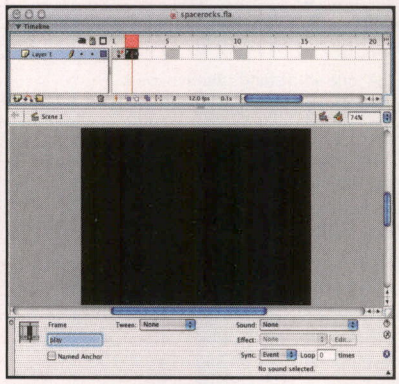
6 With the button selected, press F9 to create a script for it. Type in the simple script shown and close the script window. When the user presses this button, the movie will advance to the 'play' frame.

Part 2: Design your space rocks

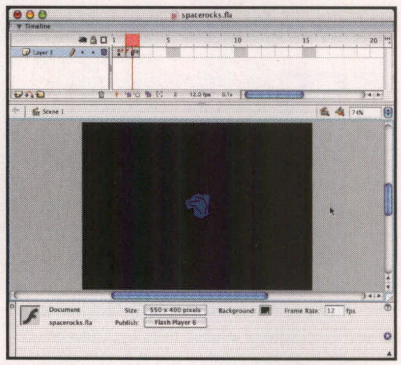
Make the rocks that you'll destroy later…

More rocks

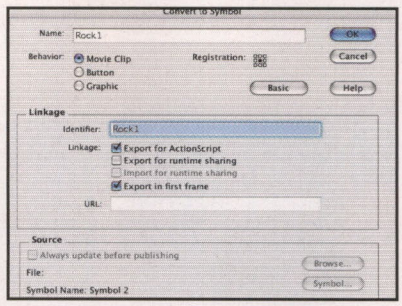
You can create more than three rocks if you want more variety. Just change the line in the code where Math.Random is multiplied by 3. For instance, if you have five rocks, change the multiplier to 5. Be sure to name the rocks 'Rock1', 'Rock2', and so on when creating the Library elements.



1 Select the second frame in the main timeline. Press F7 to insert a new keyframe there. Use the Property Inspector to name this frame 'play'.



2 In the middle of the screen, draw a rock. Make it about 60 pixels in diameter. Use whatever style you wish, from hand-drawn vector graphics to full-rendered rocks imported from a 3D program.



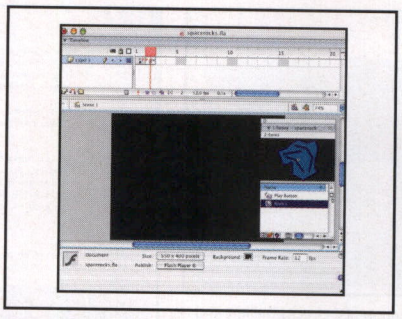
3 Select the rock drawing and press F8 to convert it into a movie clip. Make sure you select Movie Clip as the symbol's Behaviour. Name it Rock1. Also check the Export for ActionScript box. This will ensure that the movie clip is available to your code after the movie has been saved as a compressed Flash file.

Faster rocks

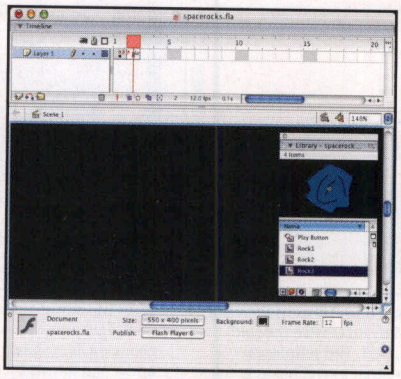
The rocks are initialised to move at a speed of four times a random number between -.5 and .5. So the speed can be between -2 and 2 in both horizontal and vertical directions. To change the potential speed of the rocks, increase or decrease the value by 4. For instance, to speed them up an average of 50 per cent, change this number to 6.

Smooth running

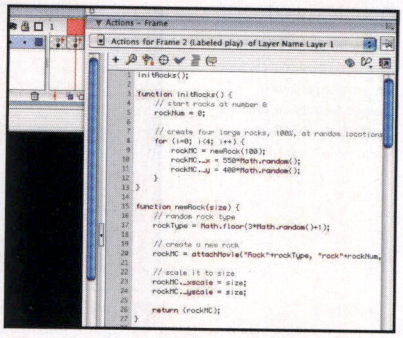
The speed of this game is somewhat independent of the framerate of the movie. This is because the setInterval command is used to make steps 50 milliseconds regardless of the framerate. However, you'll probably want to increase the framerate to some decent rate, like 30fps, to ensure smooth play.



4 Delete the movie clip from the stage. It should remain in the Library as 'Rock1' even though it's no longer on the stage anywhere. Because you selected Export for ActionScript when creating the movie clip, it'll still be included in the compressed Flash file and we can reference it with our ActionScript code.

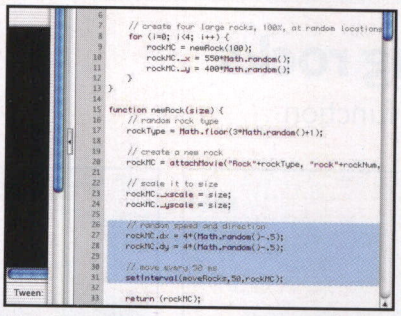


5 Create two other rocks named 'Rock2' and 'Rock3' by repeating steps 1-3 of Part 2. Make sure you check the Export for ActionScript box for both, and delete both from the stage.



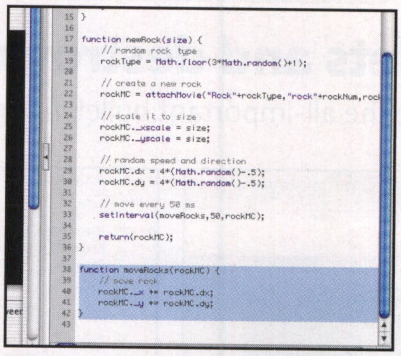
```
function initRocks() {
    // start rocks at number 0
    rockNum = 0;

    // create four large rocks, 100%, at random locations
    for (i=0; i<4; i++) {
        rockMC = newRock(100);
        rockMC._x = 550*Math.random();
        rockMC._y = 400*Math.random();
    }
}

function newRock(size) {
    // random rock type
    rockType = Math.floor(3*Math.random()+1);

    // create a new rock
    rockMC = attachMovie("Rock"+rockType, "rock"+rockNum,

    // scale it to size
    rockMC._xscale = size;
    rockMC._yscale = size;

    return(rockMC);
}
```

6 Select the second frame in the timeline and press F9 to open the script window. Add the script shown to the frame. These two functions will create four rocks in random positions on the screen. Test the movie by choosing Control>Test Movie. You should see four rocks appear at random locations.



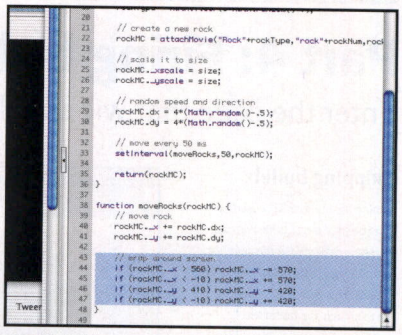
```
    // create four large rocks, 100%, at random locations
    for (i=0; i<4; i++) {
        rockMC = newRock(100);
        rockMC._x = 550*Math.random();
        rockMC._y = 400*Math.random();
    }
}

function newRock(size) {
    // random rock type
    rockType = Math.floor(3*Math.random()+1);

    // create a new rock
    rockMC = attachMovie("Rock"+rockType, "rock"+rockNum,

    // scale it to size
    rockMC._xscale = size;
    rockMC._yscale = size;

    // random speed and direction
    rockMC.dx = 4*(Math.random()-.5);
    rockMC.dy = 4*(Math.random()-.5);

    // move every 50 ms
    setInterval(moveRocks,50,rockMC);

    return(rockMC);
}
```

7 Open the frame script again and add the new code as shown in the selected area. This adds the properties dx and dy to the movie clips. This is the horizontal and vertical speed of the rock. The setInterval command will call the function moveRocks every 50 milliseconds to move the rocks that amount.



```
}

function newRock(size) {
    // random rock type
    rockType = Math.floor(3*Math.random()+1);

    // create a new rock
    rockMC = attachMovie("Rock"+rockType,"rock"+rockNum,rock

    // scale it to size
    rockMC._xscale = size;
    rockMC._yscale = size;

    // random speed and direction
    rockMC.dx = 4*(Math.random()-.5);
    rockMC.dy = 4*(Math.random()-.5);

    // move every 50 ms
    setInterval(moveRocks,50,rockMC);

    return(rockMC);
}

function moveRocks(rockMC) {
    // move rock
    rockMC._x += rockMC.dx;
    rockMC._y += rockMC.dy;
}
```

8 Add a new function – moveRocks. The moveRocks function will move a rock horizontally and vertically according to its dx and dy properties. Test the movie again to see how the rocks move.

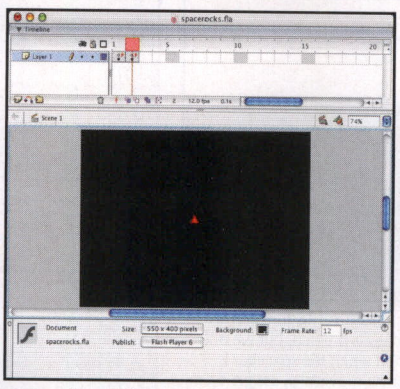


```
    // create a new rock
    rockMC = attachMovie("Rock"+rockType,"rock"+rockNum,rock

    // scale it to size
    rockMC._xscale = size;
    rockMC._yscale = size;

    // random speed and direction
    rockMC.dx = 4*(Math.random()-.5);
    rockMC.dy = 4*(Math.random()-.5);

    // move every 50 ms
    setInterval(moveRocks,50,rockMC);

    return(rockMC);
}

function moveRocks(rockMC) {
    // move rock
    rockMC._x += rockMC.dx;
    rockMC._y += rockMC.dy;

    // wrap around screen
    if (rockMC._x > 560) rockMC._x -= 570;
    if (rockMC._x < -10) rockMC._x += 570;
    if (rockMC._y > 410) rockMC._y -= 420;
    if (rockMC._y < -10) rockMC._y += 420;
}
```

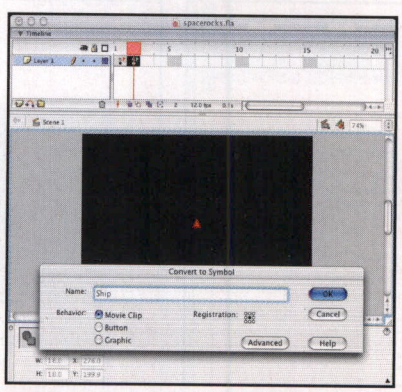
9 Add the new code to the moveRocks function. This will check the rock's new location against the edges of the screen and wrap it around to the other side if necessary. It uses the edges of the screen plus 10 pixels to make the wrap a little smoother. Test the movie again to see the rocks wrap.

Part 3: Creating the ship

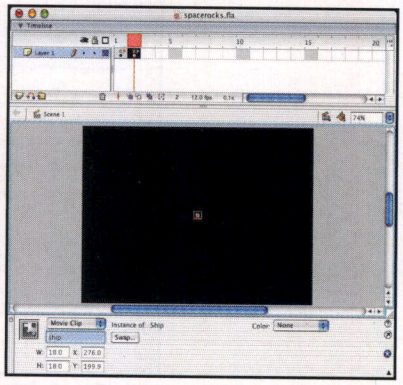
Build a spaceship and make it move…



1 At the middle of the screen, draw a small spaceship. Make sure that the front of the spaceship is pointing up.



2 Select the ship and press F8 to convert it to a movie clip. Make sure you select Movie Clip as its behaviour. Name it Ship.



3 Leave this movie clip on the stage. Select it and use the Property Inspector to name this instance of the movie clip 'ship'.

Creating the ship continued...

Double speed

To make the ship rotate faster or slower, alter the amount by which the _rotation property is changed. For instance, to double the speed of rotation, change the number to 10 instead of 5.

```
    return(rockMC);
}

function moveRocks(rockMC) {
    // move rock
    rockMC._x += rockMC.dx;
    rockMC._y += rockMC.dy;

    // wrap around screen
    if (rockMC._x > 560) rockMC._x -= 570;
    if (rockMC._x < -10) rockMC._x += 570;
    if (rockMC._y > 410) rockMC._y -= 420;
    if (rockMC._y < -10) rockMC._y += 420;
}

function initShip() {
    // start still
    ship.dx = 0;
    ship.dy = 0;

    // move every 50 ms
    setInterval(moveShip,50);
}
```

4 Add the initShip function. This sets the velocity of the ship, dx and dy, to 0 since the ship should start off standing still. The setInterval command will make sure that moveShip gets called once every 50 milliseconds to move the ship. Add the call to initShip(); right under the call to initRocks(); at the start of the script.

```
    setInterval(moveShip,50);
}

function moveShip() {
    // rotate ship left and right
    if (Key.isDown(Key.LEFT)) {
        ship._rotation -= 5;
    }
    if (Key.isDown(Key.RIGHT)) {
        ship._rotation += 5;
    }

    // thrust forward
    if (Key.isDown(Key.UP)) {
        shipAngle = 2.0*Math.PI*(ship._rotation-90)/360.0;
        ship.dx += Math.cos(shipAngle);
        ship.dy += Math.sin(shipAngle);
    }

    // move ship
    ship._x += ship.dx;
    ship._y += ship.dy;
}
```

5 The moveShip function first checks the left and right arrow keys and rotates the ship by changing its _rotation property. It then checks the up arrow and thrusts the ship forward if it's pressed. It does this by using Math.cos and Math.sin to convert the rotation of the ship into x and y components. It then moves the ship according to dx and dy.

```
}

function moveShip() {
    // rotate ship left and right
    if (Key.isDown(Key.LEFT)) {
        ship._rotation -= 5;
    }
    if (Key.isDown(Key.RIGHT)) {
        ship._rotation += 5;
    }

    // thrust forward
    if (Key.isDown(Key.UP)) {
        shipAngle = 2.0*Math.PI*(ship._rotation-90)/360.0;
        ship.dx += Math.cos(shipAngle);
        ship.dy += Math.sin(shipAngle);
    }

    // move ship
    ship._x += ship.dx;
    ship._y += ship.dy;

    // wrap around screen
    if (ship._x > 550) ship._x -= 550;
    if (ship._x < 0) ship._x += 550;
    if (ship._y > 400) ship._y -= 400;
    if (ship._y < 0) ship._y += 400;
}
```

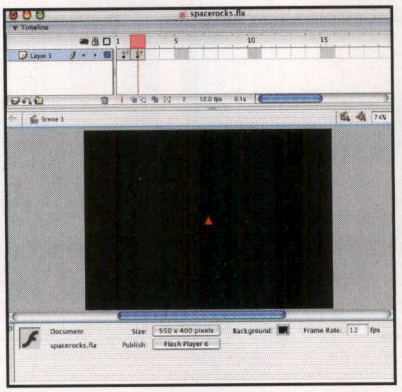
6 Add the four lines of code shown above so that the ship will also wrap around the four edges of the screen.

Part 4: Firing bullets and destroying rocks

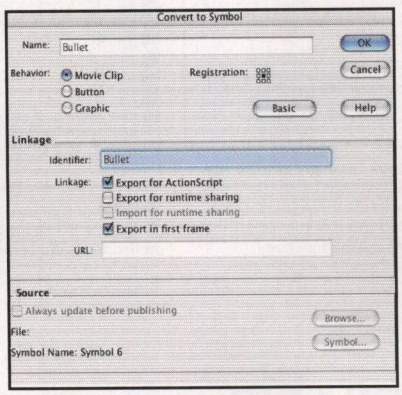
Enter the code that will enable the all-important bullet-firing function...

Skipping bullets

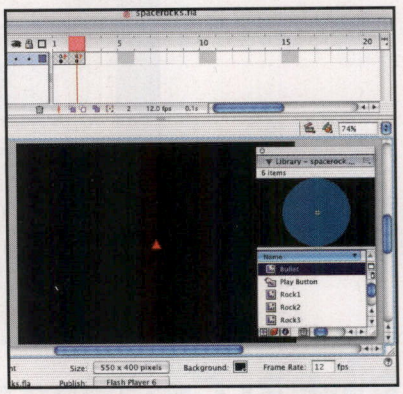
Notice the number 10 is used for the speed of the bullets. You can increase or decrease this number to speed up, or slow down, the bullets. However, if you speed it up too much, the bullet will appear to skip across the screen. In addition, this skipping will mean that it might skip right over a rock and miss it.



1 Create a small circle on the stage to act as the bullets which the ship will fire. Make the dot only a few pixels in diameter.



2 Select the dot and press F8 to convert it to a symbol. Give it a Movie Clip Behaviour. Name it Bullet and select Export for ActionScript.



3 Delete the bullet from the stage. This will leave it in the Library. We can refer to this Library member as 'bullet' in our code.

```
initShip();

// listen for key presses
spacebarListener = new Object();
spacebarListener.onKeyDown = fireBullet;
Key.addListener(spacebarListener);

function initRocks() {
    // start rocks at number 0
    rockNum = 0;

    // create four large rocks, 100%, at random locations
    for (i=0; i<4; i++) {
        rockMC = newRock(100);
        rockMC._x = 550*Math.random();
        rockMC._y = 400*Math.random();
    }
}

function newRock(size) {
    // random rock type
    rockType = Math.floor(3*Math.random()+1);

    // create a new rock
    rockMC = attachMovie("Rock"+rockType, "rock"+rockNum,

    // scale it to size
    rockMC._xscale = size;
    rockMC._yscale = size;
```

4 Add the code shown to the very beginning of the script. This makes the movie listen for key presses and pass them on to the fireBullet function.

```
    ship._x += ship.dx;
    ship._y += ship.dy;

    // wrap around screen
    if (ship._x > 550) ship._x -= 550;
    if (ship._x < 0) ship._x += 550;
    if (ship._y > 400) ship._y -= 400;
    if (ship._y < 0) ship._y += 400;
}

function fireBullet() {
    if (Key.getAscii() == 32) {

        // create new bullet at level 1000+bulletNum
        bulletMC = attachMovie("Bullet","bullet"+bulletNum,1000+bulletNum++);

        // start at center of ship
        bulletMC._x = ship._x;
        bulletMC._y = ship._y;

        // direction of ship
        shipAngle = 2*Math.PI*(ship._rotation-90)/360;
        bulletMC.dx = Math.cos(shipAngle);
        bulletMC.dy = Math.sin(shipAngle);

        // attach movement function
        bulletMC.moveBullet = moveBulletFunction;
        setInterval(bulletMC,"moveBullet",50,bulletMC)
    }
}
```

5 Add the fireBullet function. This function is called whenever a key is pressed, but we only want to perform an action when the spacebar is pressed. It will create a new movie clip, assign it a dx and dy according to the ship's orientation, and set up moveBulletFunction to be called from inside the new movie clip every 50 milliseconds.

```
    // wrap around screen
    if (ship._x > 550) ship._x -= 550;
    if (ship._x < 0) ship._x += 550;
    if (ship._y > 400) ship._y -= 400;
    if (ship._y < 0) ship._y += 400;
}

function fireBullet() {
    if (Key.getAscii() == 32) {

        // create new bullet at level 1000+bulletNum
        bulletMC = attachMovie("Bullet","bullet"+bulletNum,1000+bul

        // start at center of ship
        bulletMC._x = ship._x;
        bulletMC._y = ship._y;

        // direction of ship
        shipAngle = 2*Math.PI*(ship._rotation-90)/360;
        bulletMC.dx = Math.cos(shipAngle);
        bulletMC.dy = Math.sin(shipAngle);

        // attach movement function
        bulletMC.moveBullet = moveBulletFunction;
        setInterval(bulletMC,"moveBullet",50,bulletMC)
    }
}

function moveBulletFunction() {
    // move bullet
    bulletSpeed = 10.0;
    this._x += bulletSpeed*this.dx;
    this._y += bulletSpeed*this.dy;
}
```

6 The moveBulletFunction will push the bullet along according to its own dx and dy properties.

Multiple lives

You may want to give the player multiple chances, or lives, so they can get hit by a rock three times before the game is over. This gets tricky as you need to reset the ship and wait for all rocks to clear the centre area before resuming the game. It'll add quite a bit more code to the game, but expert ActionScript programmers should be able to do it.

```
}
function fireBullet() {
    if (Key.getAscii() == 32) {
        // create new bullet at level 1000+bulletNum
        bulletMC = attachMovie("Bullet", "bullet"+bulletNum, 1000+bulle
        // start at center of ship
        bulletMC._x = ship._x;
        bulletMC._y = ship._y;
        // direction of ship
        shipAngle = 2*Math.PI*(ship._rotation-90)/360;
        bulletMC.dx = Math.cos(shipAngle);
        bulletMC.dy = Math.sin(shipAngle);
        // attach movement function
        bulletMC.moveBullet = moveBulletFunction;
        setInterval(bulletMC, "moveBullet", 50, bulletMC);
    }
}

function moveBulletFunction() {
    // move bullet
    bulletSpeed = 10.0;
    this._x += bulletSpeed*this.dx;
    this._y += bulletSpeed*this.dy;

    // see if reached end of screen
    if (this._x > 550) this.removeMovieClip();
    if (this._x < 0) this.removeMovieClip();
    if (this._y > 400) this.removeMovieClip();
    if (this._y < 0) this.removeMovieClip();
}
```

7 This new section of code tests to see if the bullet has reached the edge of the screen. It will then remove itself. So bullets won't wrap, but instead only travel until they hit the edge of the screen.

```
// listen for key presses
spacebarListener = new Object();
spacebarListener.onKeyDown = fireBullet;
Key.addListener(spacebarListener);

function initRocks() {
    rocksArray = new Array();

    // start rocks at number 0
    rockNum = 0;

    // create four large rocks, 100%, at random locations
    for (i=0; i<4; i++) {
        rockMC = newRock(100);
        rockMC._x = 550*Math.random();
        rockMC._y = 400*Math.random();
    }
}

function newRock(size) {
    // random rock type
    rockType = Math.floor(3*Math.random()+1);

    // create a new rock
    rockMC = attachMovie("Rock"+rockType, "rock"+rockNum,
```

8 To determine if a bullet and a rock meet, you first have to start keeping tabs on the rocks. Right now they're all independent objects. You need to create an array and store a reference to each rock in the array.

```
function newRock(size) {
    // random rock type
    rockType = Math.floor(3*Math.random()+1);

    // create a new rock
    rockMC = attachMovie("Rock"+rockType, "rock"+rockNum,

    // scale it to size
    rockMC._xscale = size;
    rockMC._yscale = size;

    // random speed and direction
    rockMC.dx = 4*(Math.random()-.5);
    rockMC.dy = 4*(Math.random()-.5);

    // move every 50 ms
    setInterval(moveRocks,50,rockMC);

    // store in array
    rocksArray.push(rockMC);

    return (rockMC);
}

function moveRocks(rockMC) {
    // move rock
    rockMC._x += rockMC.dx;
    rockMC._y += rockMC.dy;

    // wrap around screen
    if (rockMC._x>550) rockMC._x -= 570;
```

9 The above code will add each rock to the array as it's created.

```
function moveBulletFunction() {
    // move bullet
    bulletSpeed = 10.0;
    this._x += bulletSpeed*this.dx;
    this._y += bulletSpeed*this.dy;

    // see if reached end of screen
    if (this._x > 550) this.removeMovieClip();
    if (this._x < 0) this.removeMovieClip();
    if (this._y > 400) this.removeMovieClip();
    if (this._y < 0) this.removeMovieClip();

    // loop through rocks to see if bullet hit any
    for (var i = _root.rocksArray.length-1; i>=0; i--) {
        if (_root.rocksArray[i].hitTest(this._x, this._y)) {

            // remove bullet
            this.removeMovieClip();

            // deal with rock destruction
            _root.hitRock(_root.rocksArray[i]);

            // remove rock from array
            _root.rocksArray.splice(i, 1);
        }
    }
}
```

10 This code, added to the moveBulletFunction function will check each rock in the array to see if the bullet has collided with it. If so, the bullet removes itself, it calls the hitRock function to let the rock deal with its own destruction, and the rock is removed from the array.

```
            // remove rock from array
            _root.rocksArray.splice(i, 1);
        }
    }
}

function hitRock(rockMC) {
    if (rockMC._xscale > 25) {
        // if rock is large enough, split in two
        for (var i=0;i<2;i++) {
            // new rocks are half the size
            // starting at old location
            newRockMC = newRock(rockMC._xscale/2);
            newRockMC._x = rockMC._x;
            newRockMC._y = rockMC._y;
        }
    }

    // remove old rock
    rockMC.removeMovieClip();
}
```

11 The hitRock function will check the rock's scale. If it's greater than 25 per cent, it'll create two new rocks at half the original rock's scale. So a rock at 100 per cent will split to two 50 per cent rocks; a rock at 50 per cent will split into two 25 per cent rocks; and a 25 per cent rock will just disappear. The old rock movie clip will be removed. You can test the movie now to see the bullets in action.

```
    // move ship
    ship._x += ship.dx;
    ship._y += ship.dy;

    // wrap around screen
    if (ship._x > 550) ship._x -= 550;
    if (ship._x < 0) ship._x += 550;
    if (ship._y > 400) ship._y -= 400;
    if (ship._y < 0) ship._y += 400;

    // see if ship hit rock
    for(var i in rocksArray) {
        if (rocksArray[i].hitTest(ship._x,ship._y)) {
            gotoAndStop("lose");
        }
    }
}

function fireBullet() {
    if (Key.getAscii() == 32) {

        // create new bullet at level 1000+bulletNum
        bulletMC = attachMovie("Bullet", "bullet"+bulletNum,

        // start at center of ship
        bulletMC._x = ship._x;
        bulletMC._y = ship._y;

        // direction of ship
```

12 The new code added to moveShip will loop through the array of rocks and check to see if any is too close to the ship. If so, the movie jumps to the "lose" frame.

Part 5: Prepare to die

Making sure the player doesn't get killed immediately…

```
    if (this._y < 0) this.removeMovieClip();

    // loop through rocks to see if bullet hit any
    for (var i = _root.rocksArray.length-1; i>=0; i--) {
        if (_root.rocksArray[i].hitTest(this._x, this._y)) {

            // remove bullet
            this.removeMovieClip();

            // deal with rock destruction
            _root.hitRock(_root.rocksArray[i]);

            // remove rock from array
            _root.rocksArray.splice(i, 1);

            // see if all rocks are gone
            if (_root.rocksArray.length == 0) {
                gotoAndStop("win");
            }
        }
    }
}

function hitRock(rockMC) {
    if (rockMC._xscale > 25) {
        // if rock is large enough, split in two
```

1 This new code added to the moveBulletFunction function will check to see if there are any rocks left in the array. If not, then the player has destroyed them all. The game jumps to the "win" frame.

```
initRocks();
initShip();

// listen for key presses
spacebarListener = new Object();
spacebarListener.onKeyDown = fireBullet;
Key.addListener(spacebarListener);

// start time for game
gameStartTime = getTimer();

function initRocks() {
    rocksArray = new Array();

    // start rocks at number 0
    rockNum = 0;

    // create four large rocks, 100%, at random locations
    for (i=0; i<4; i++) {
        rockMC = newRock(100);
        rockMC._x = 550*Math.random();
        rockMC._y = 400*Math.random();
    }
}

function newRock(size) {
    // random rock type
    rockType = Math.floor(3*Math.random()+1);

    // create a new rock
```

2 You need to avoid the case where the game starts with a rock already on top of the ship because this gives the player no hope of survival. So record the time that the game started.

```
function moveShip() {
    // rotate ship left and right
    if (Key.isDown(Key.LEFT)) {
        ship._rotation -= 5;
    }
    if (Key.isDown(Key.RIGHT)) {
        ship._rotation += 5;
    }
    // thrust forward

    if (Key.isDown(Key.UP)) {
        shipAngle = 2.0*Math.PI*(ship._rotation-90)/360.0;
        ship.dx += Math.cos(shipAngle);
        ship.dy += Math.sin(shipAngle);
    }

    // move ship
    ship._x += ship.dx;
    ship._y += ship.dy;

    // wrap around screen
    if (ship._x > 550) ship._x -= 550;
    if (ship._x < 0) ship._x += 550;
    if (ship._y > 400) ship._y -= 400;
    if (ship._y < 0) ship._y += 400;

    // see if ship hit rock
    if (getTimer() > gameStartTime + 3000) {
        for(var i in rocksArray) {
            if (rocksArray[i].hitTest(ship._x,ship._y)) {
                gotoAndStop("lose");
            }
```

3 You need to make sure that the game has been going on for at least three seconds before allowing the player to die. ›

Part 6: Game over screen and score card

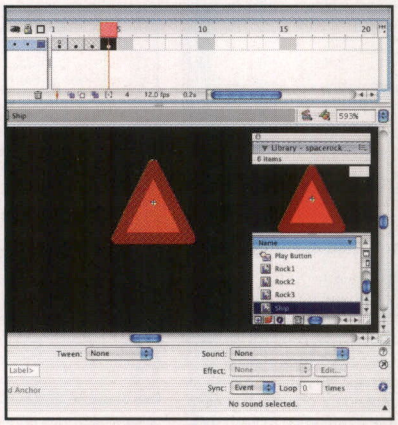
Adding animated thrusters, scoring and closing screens…

Additional frames

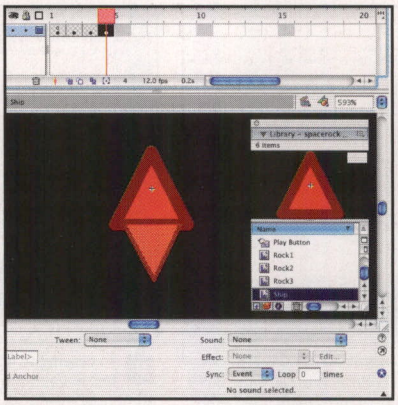
You can add a lot more than three frames if you want to. As long as no scripts are placed in any of these new frames, the movie clip will animate to the end and loop back to frame 1.

Multiple levels

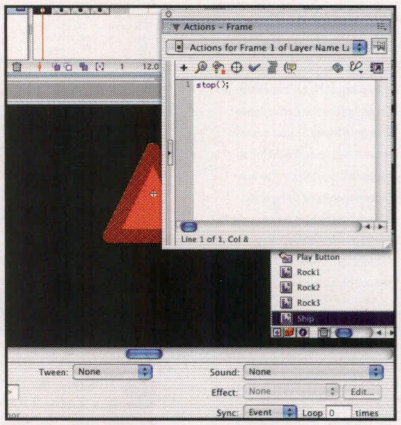
You may choose to create a game with multiple levels. This can get tricky, but it basically comes down to not resetting the score after a win, and also increasing the speed or number of rocks when the new level begins. This will require expert ActionScript skills.



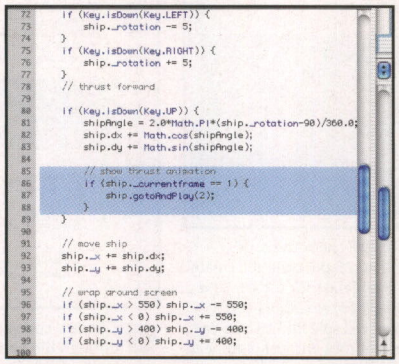
1 Open the Library panel and double-click on the Ship movie clip. Select the second frame and press F6 to duplicate the first keyframe there. Do this three times to create frame 2, 3 and 4.



2 Go to each of the frames 2, 3 and 4, and add flames to the bottom of the ship. Each of these three frames should have differently shaped flames. When the thrusters fire, the ship will animate between frames 2, 3 and 4 very quickly.

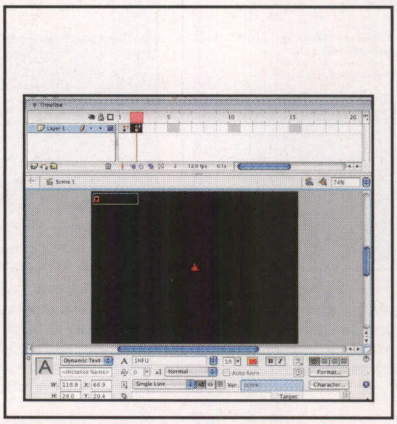


3 Choose the first frame and press F9 to bring up the script panel. Add the 'stop();' script – without this, the ship movie clip will continuously animate from the very start.

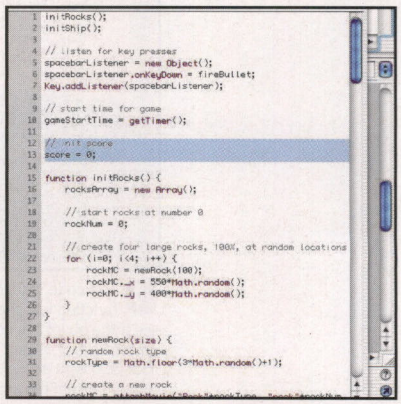


```
    if (Key.isDown(Key.LEFT)) {
        ship._rotation -= 5;
    }
    if (Key.isDown(Key.RIGHT)) {
        ship._rotation += 5;
    }
    // thrust forward

    if (Key.isDown(Key.UP)) {
        shipAngle = 2.0*Math.PI*(ship._rotation-90)/360.0;
        ship.dx += Math.cos(shipAngle);
        ship.dy += Math.sin(shipAngle);

        // show thrust animation
        if (ship._currentframe == 1) {
            ship.gotoAndPlay(2);
        }
    }

    // move ship
    ship._x += ship.dx;
    ship._y += ship.dy;

    // wrap around screen
    if (ship._x > 550) ship._x -= 550;
    if (ship._x < 0) ship._x += 550;
    if (ship._y > 400) ship._y -= 400;
    if (ship._y < 0) ship._y += 400;
```

4 Add this code to the moveShip function. This checks to see whether the ship is already animating, and will only jump to frame 2 if it's not. Once on frame 2, the animation will automatically continue to frames 3 and 4 and then back to 1. At that point, if the up arrow is still down, the animation will start again.

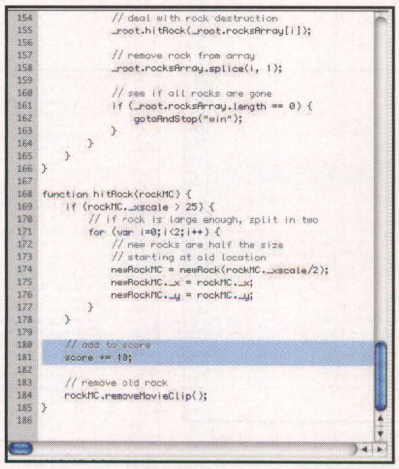


5 Use the Text tool to create a text area at the top-left side of the screen. Start this area with the number 0. Use the Property Inspector to set its type to Dynamic Text and its 'Var' to 'score'.



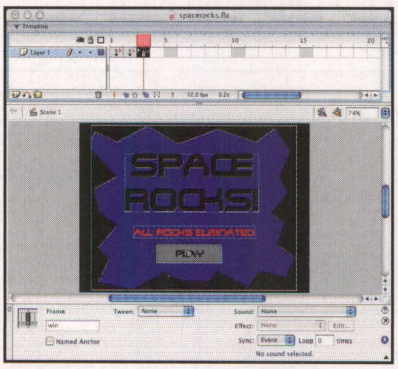
```
initRocks();
initShip();

// listen for key presses
spacebarListener = new Object();
spacebarListener.onKeyDown = fireBullet;
Key.addListener(spacebarListener);

// start time for game
gameStartTime = getTimer();

// init score
score = 0;

function initRocks() {
    rocksArray = new Array();

    // start rocks at number 0
    rockNum = 0;

    // create four large rocks, 100%, at random locations
    for (i=0; i<4; i++) {
        rockMC = newRock(100);
        rockMC._x = 550*Math.random();
        rockMC._y = 400*Math.random();
    }
}

function newRock(size) {
    // random rock type
    rockType = Math.floor(3*Math.random())+1;

    // create a new rock
```

6 Add this code to start the score of the game at 0. The variable 'score' and the text field created in the previous step are linked, so any changes to the variable score will be immediately shown in the text field.

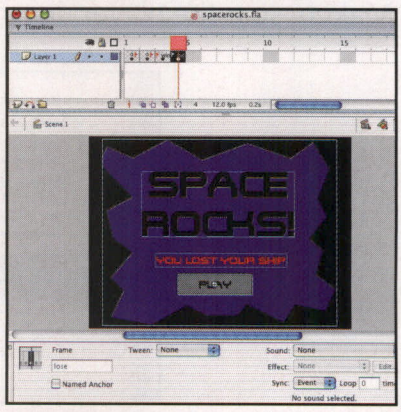


```
            // deal with rock destruction
            _root.hitRock(_root.rocksArray[i]);

            // remove rock from array
            _root.rocksArray.splice(i, 1);

            // see if all rocks are gone
            if (_root.rocksArray.length == 0) {
                gotoAndStop("win");
            }
        }
    }
}

function hitRock(rockMC) {
    if (rockMC._xscale > 25) {
        // if rock is large enough, split in two
        for (var i=0;i<2;i++) {
            // new rocks are half the size
            // starting at old location
            newRockMC = newRock(rockMC._xscale/2);
            newRockMC._x = rockMC._x;
            newRockMC._y = rockMC._y;
        }
    }

    // add to score
    score += 10;

    // remove old rock
    rockMC.removeMovieClip();
}
```

7 In the hitRock function, award the player ten points each time a rock has been hit.



8 Back in the main timeline, select the third frame and press F7 to create a new keyframe there. Use the Property Inspector to name this frame 'win'. Create graphics for this 'win' frame. We re-used the graphics from the first frame but with some additional text. Also include the Play button so that the user can start a new game.

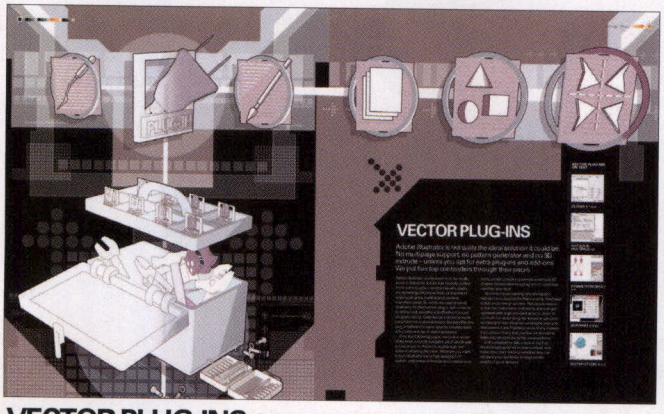


9 Now select the fourth frame in the main timeline and press F7 to create a new keyframe there. Name this frame 'lose' and add graphics to it. Include the Play button again so the user can start a new game. ca-s

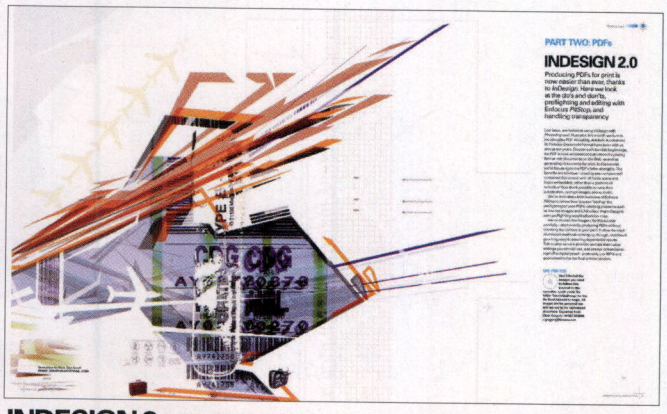
WINTER ISSUE ON SALE NOW

SETTING UP IN 3D OUR INDUSTRY REPORT ON WHAT IT TAKES TO FLY SOLO IN THE DOG-EAT-DOG WORLD OF 3D

DIRECTOR MX IN-DEPTH PREVIEW OF THE LEADING MULTIMEDIA APP IN OUR 21-PAGE REVIEW SECTION



VECTOR PLUG-INS OUR DEFINITIVE GROUP TEST ON THE LATEST AND BEST PLUG-INS FOR ILLUSTRATOR



INDESIGN 2 PDF CREATION AND PREFLIGHTING USING ADOBE'S GROWING GRAPHIC DESIGN PACKAGE

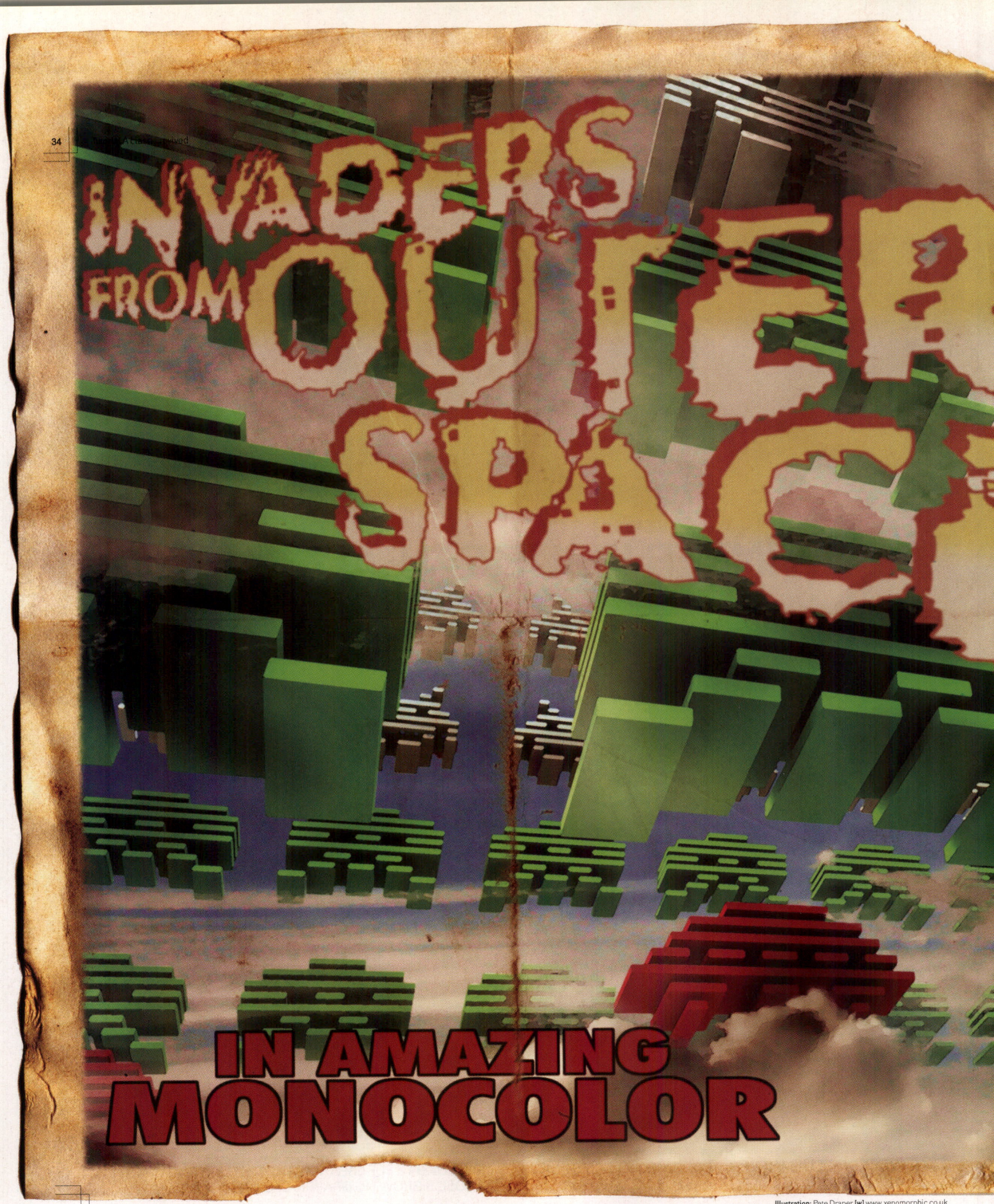


Illustration: Pete Draper **[w]** www.xenomorphic.co.uk

Space Invaders

With the *Flash* games market thriving like never before we're going back to learn from the roots – a classic revived in 45 steps!

Let's see what we can learn from recreating the mother of all space games in the modern environment of *Flash MX*. The concept is simple – evil aliens move faster and faster down the screen towards a defending ground ship. However, programming *Space Invaders* will take you through some of the most essential development techniques that you'll come to use, whatever action game you want to create. We'll show you how to clone a single alien into a full armada of evil shooting creatures; how to detect collisions between different objects and time them with explosions and sounds; how to add user key control; and how to count scores. We'll finish it all off by setting up a score board.

To follow this tutorial you'll need to have a good understanding of how to use the *Flash MX* environment and at least a basic understanding of ActionScripting using dotted syntax. If up until now you've only briefly played around with ActionScript and have never used it in Expert mode, you should prepare yourself for a bumpy ride. Still, these could be your first vital steps into the powerful world of ActionScripting.

Before starting with the 45 steps below, have a good look at the finished product which is supplied on the CD. Play through the game until you're familiar with all the different events that occur in it, as well as all the different objects that exist and how they behave. Happy reading! >

Tutorial and code by Niklas Alvaeus
[e] helloniklas@yahoo.com
[w] www.helloniklas.com
Graphics and animation by Suman Joshi
[e] suman@zoom.co.uk

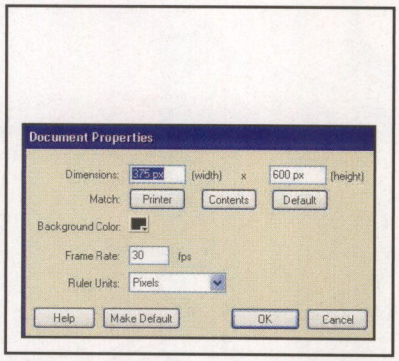
Files on disc
The files that support this tutorial can be found in the Tutorial\gm_space folder on the Cover CD.

Part 1: Setting up the graphics

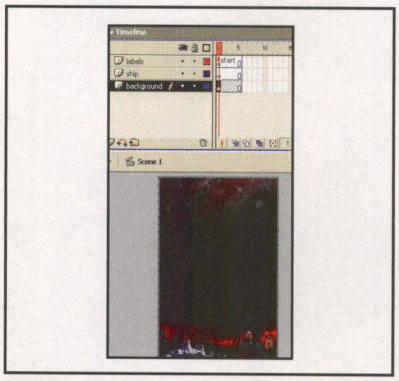
Create all the explosions, aliens and spaceship images you'll need...

Storing your ActionScripts and frame labels

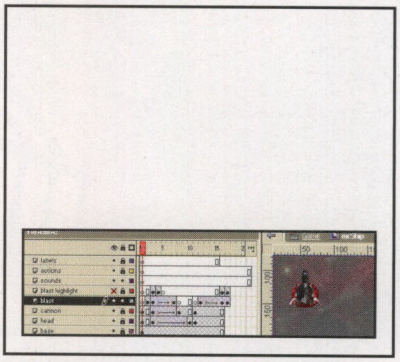
It's good practice to keep your frame labels on their own layer, separate from code and graphics, and let it stay on top of all the other layers. Separating your ActionScripts over different layers will also enhance the readability of the code and bring a clear structure to the build of the game.



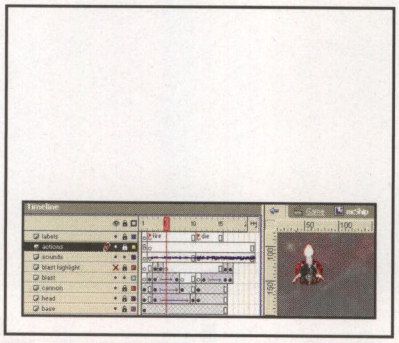
1 To start with, you'll need to launch *Flash MX* and create a new document. Begin by modifying its basic properties (Modify>Document). Change the dimensions to 375 pixels width and 600 pixels height. Now change the background colour to black and set the frame rate to 30 fps. Finally, click OK.



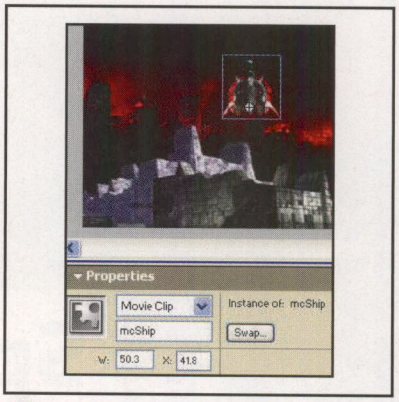
2 Insert two new layers. Name the top one Labels, the bottom one Background and the middle one Ship. With the Background layer selected, import a bitmap and place it on stage at position 0, 0. Click on the first frame in the Labels layer and set the frame label name to "start".



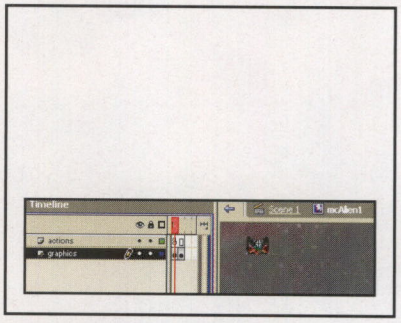
3 Create a new movie clip (Ctrl+F8) and name it mcShip. This clip will hold the graphics for the defending spaceship. Add layers for labels, actions, sounds and whatever other layers you'll need for the graphics. The ship itself will have three states: one when it's doing nothing, one when it's shooting and one when it's dying.



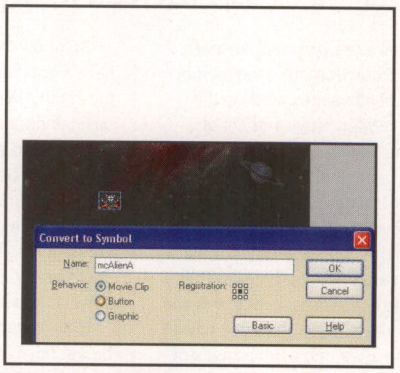
4 Inside mcShip create frame labels called "fire" and "die" on the Labels layer. Fill the frames under label "fire" with animations and sounds for the spaceship shooting, and the frames under label "die" with animations and sounds for the spaceship dying. Leave the first frame for its normal state. Create a keyframe on this first frame on the Actions layer and put the code stop() on it.



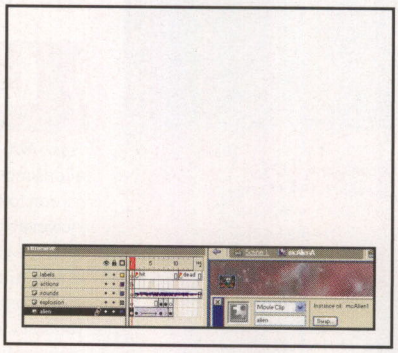
5 Go back to the main timeline and select the Ship layer. Drag the new movie clip of the ship onto the stage. Select the movie clip and give it the instance name mcShip. The space ship is now ready for action but leave it there for now.



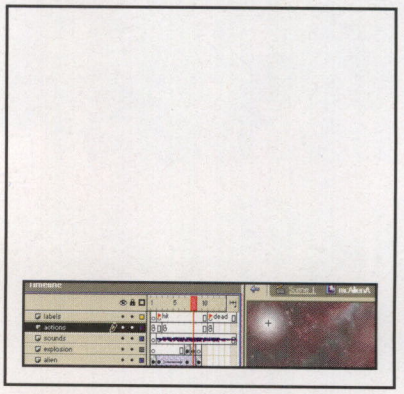
6 Create another movie clip (Ctrl+F8) called mcAlien1. Set it up with two layers, one called Actions and one called Graphics. This movie clip will have two frames to create the aliens' stepping animation that is the trademark of the classic *Space Invaders* game, the first frame showing first position and the second the other position. Put the code stop() in the first frame on the Actions layer to prevent the clip from looping.



7 Go back to the main timeline and create another layer called gfx. With the gfx layer selected, drag out the movie clip mcAlien1 from the library and onto the stage. Select the movie clip and convert it into a second nested movie clip by pressing F8. Call it mcAlienA. Double click on the new movie clip mcAlienA to get into its timeline.



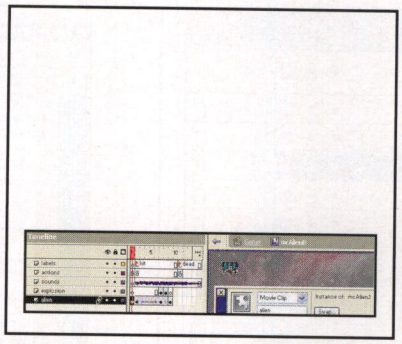
8 Like the ship we created in step 3, this alien will also need three states: one normal, one when it's been hit and one when it's dead. Create new layers for labels, actions, sounds and explosions. Leave the mcAlien1 on its own layer called Alien. Select mcAlien1 and give it the instance name Alien. We will need to reference this movie clip later on to animate between the two frames that reside in it.



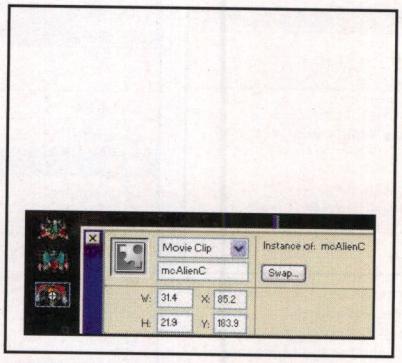
9 Create a tween of the alien exploding, starting from the frame labelled "hit". Add the sound starting from the same frame. Leave the frames empty under the frame labelled "dead". Add the code play() on the Actions layer under the "hit" frame. Add a stop() under the "dead" frame .

Masking bitmaps

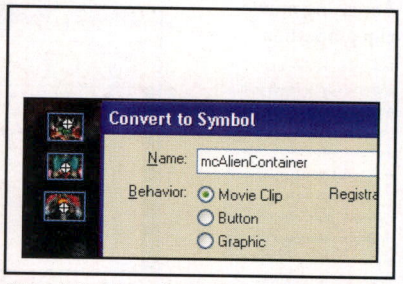
Converting the vector graphics into bitmaps can greatly increase the speed of the game. Simply take a screen grab of your objects in *Flash*, crop them down in *Photoshop* and save them as .gifs or .png files with a transparent background. Import them back into *Flash* and replace your vector graphics with the bitmap. Sometimes you get a smoother result by putting a mask around the imported bitmap instead of using the transparency in the image.



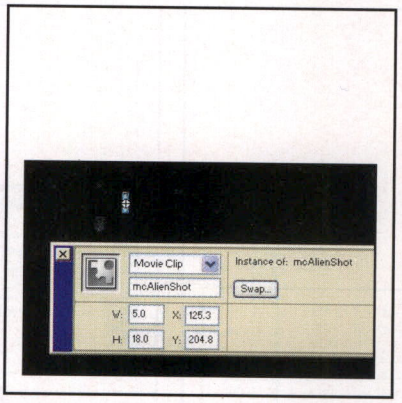
10 Repeat the process from step 6 two more times, creating a movie clip, mcAlien2, that resides inside a movie clip called mcAlienB and a third alien called mcAlien3 that resides inside mcAlienC. Don't forget to set the instance name of all of them to Alien. Since they all sit inside different movie clips they will have a unique reference path, although they all have the instance name Alien.



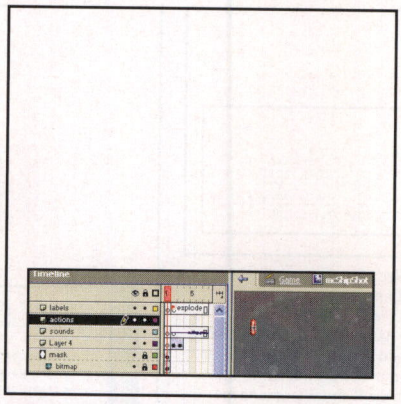
11 You should now have three aliens sitting on the layer called gfx on the main timeline. Select each one of them and give them the instance names of mcAlienA, mcAlienB and mcAlienC respectively. Note that the instance name doesn't have to be the same as that of the symbol, but it makes it easier to keep track of things.



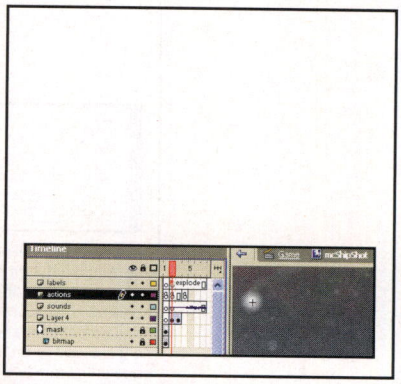
12 Select all three aliens and press F8 (Insert> Convert to Symbol). Select Movie Clip and type the name mcAlienContainer. We now have a single movie clip on the stage called mcAlienContainer which contains all three aliens. This will make it more efficient later on when we will animate all the aliens moving across the stage as we will just have to change the properties of mcAlienContainer to make all the movie clips inside it move. Select the movie clip mcAlienContainer and set its instance name to mcAlienContainer.



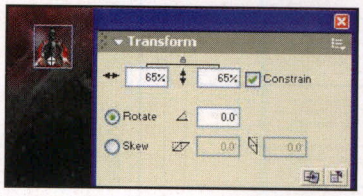
13 Create a new movie clip and name it mcAlienShot. Put some graphics resembling a bullet on its first frame. Drag it out to the main timeline on the gfx layer. Select it and give it the instance name of mcAlienShot.



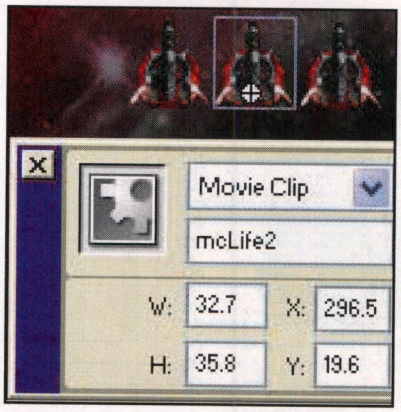
14 Create another movie clip and name it mcShipShot. Inside, create layers for labels, actions, sounds, explosions and layers for the graphics. We use a masked bitmap for this. Put the graphics of the bullet itself on the first frame.



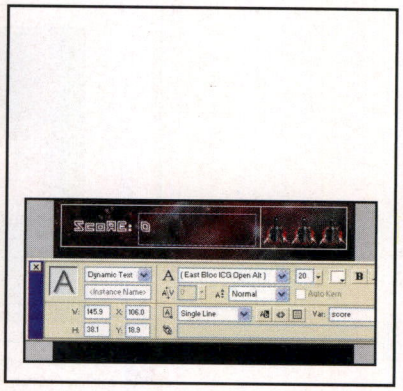
15 Create a label named Explosion on the second keyframe on the layer Labels. Put a small explosion on this key frame. Add the code stop() on the first frame on the Actions layer, play() on the frame labelled Explode and another stop() on the last frame.



16 Go back to the main timeline and create a new layer called Lives. Select your ship mcShip and press Ctrl+C to copy it. Select the layer Lives you just created and press Ctrl+V to paste in a copy of the ship. With the copied ship selected, press Ctrl+T to bring up the transform palette. Change the scale to about 65 per cent with the Constrain box ticked.



17 Make two more copies of the scaled down copy of the ship and place them next to each other towards the top right corner of the stage. Give them the instance names of mcLife1, mcLife2 and mcLife3 respectively.



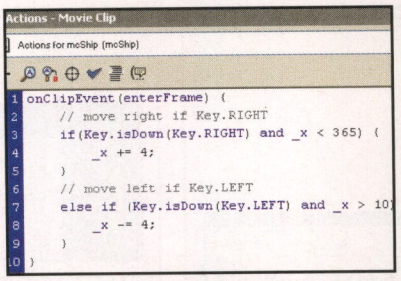
18 Create a new layer on the main timeline called Outlines and another one called Score. Draw a grid on the layer Outlines. Select the Score layer and type the text "Score", followed by an empty text box. Select the empty text box and make sure it's set to Dynamic text. Set its Var property to Score.

Part 2: Creating the game and high score table

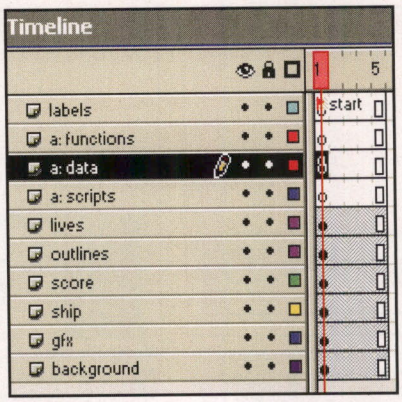
Animate your images as you recreate the classic alien attack…

Accessing movie clip properties

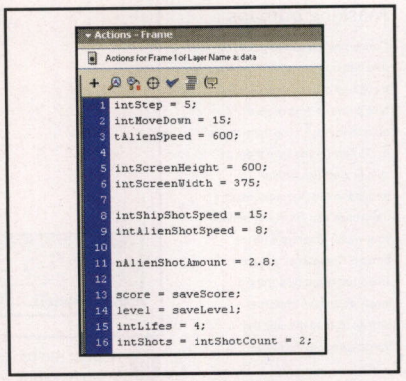
To access a movie clip's properties dynamically with ActionScripting, use the _root["myclip" + i]._x; instead of the older myClip = eval("_root.myclip" + i); myClip._x; statement. Using the older eval-style statements can in certain circumstances cause problem in *Flash MX*, and it's much more hassle anyway.



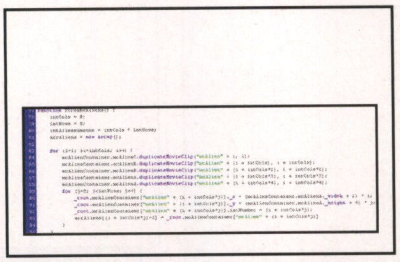
1 Select the mcShip movie clip. Press F9 to bring up the Actions panel and add an onClipEvent(enterFrame). The code inside the brackets will run as frequently as the frame rate of the movie. Check if the right arrow key is pressed and that the x position of the movie clip mcShip is less than 365 to prevent it going off stage. If it does, we increase the x position of mcShip by four pixels. If the left arrow key is pressed and the position is more than 10, decrease the x position. We've now got key control of mcShip.



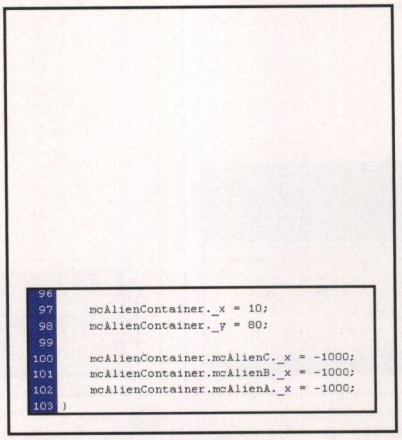
2 Create three new layers on the main timeline called a: functions, a: data and a: scripts. These layers will hold functions, data and scripts calling the functions. Select the first keyframe on the a: data layer and press F9 to open the Actions panel.



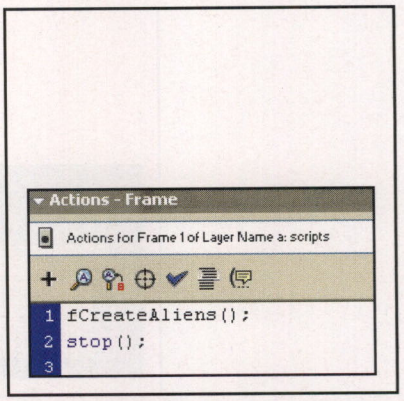
3 Initialise the variables we will use. Place the a: scripts layer beneath the a: data and a: functions layers to make sure the functions and data are loaded before we make any use of them from the a: script layer. Select the first frame on the layer a: functions. We'll keep our functions here.



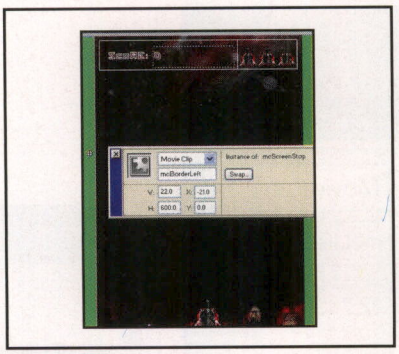
4 Add a function called fCreateAliens() which will duplicate the three aliens (mcAlienA, mcAlienB and mcAlienC) into a whole armada of 49 aliens with the instance names mcAlien1, mcAlien2, mcAlien3 and so on, up to mcAlien49. They will all reside in the movie clip mcAlienContainer, so their full reference path is mcAlienContainer.mcAlien1, mcAlienContainer.mcAlien2 and so on. Save a reference to each one of them inside an array called arrAliens. Set a variable called intNumber in every one of them saying which one of the 49 aliens they actually are.



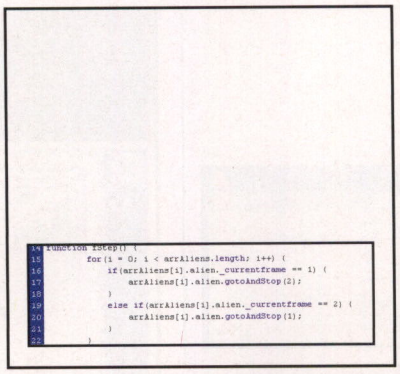
5 Position the mcAlienContainer towards the top left of the stage and hide the movie clips of our original aliens by putting them far off the stage.



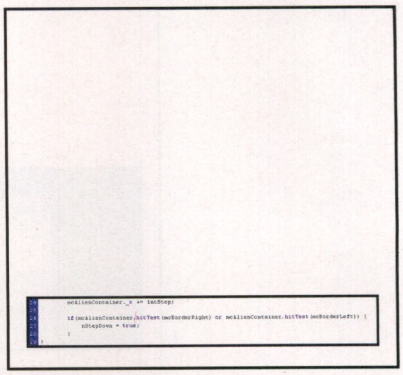
6 Click on the first frame on the a: scripts layer and add a call to the fCreateAliens function. Put a stop() afterwards to halt the play head on the first frame. Running the *Flash* file should produce a group of 49 aliens sitting at the top of the stage.



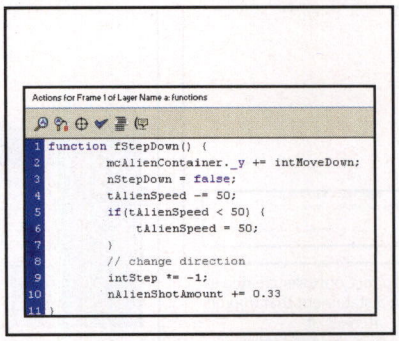
7 Create a new movie clip (Ctrl+F8) and name it mcScreenStop. Draw a solid shape about 20 pixels wide and as high as the whole stage (600 pixels) on its first frame. Go back to the main timeline and drag two instances of the new movie clip out on the stage. Give them the instance name of mcBorderLeft and mcBorderRight and put them just outside the edges of the stage.



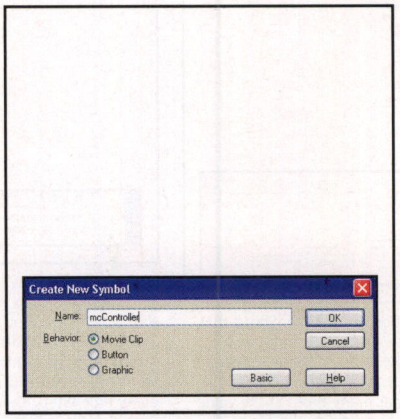
8 Click on the first frame in the a: functions layer and press F9 to bring up the Actions panel. Add a new function called fStep(). Loop through all the aliens stored in the array arrAliens and check what frame the movie clip alien is on (see step 8 in part 1). Change the frame of the movie clip alien to create the stepping animation.



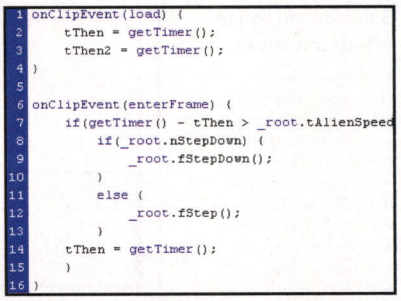
9 Change the x position of all the aliens by adding the variable intStep to the x position of movie clip mcAlienContainer. Check if mcAlienContainer reached either side of the screen by seeing if it has collided with mcBorderRight or mcBorderLeft. Set the variable nStepDown to true if this has happened.



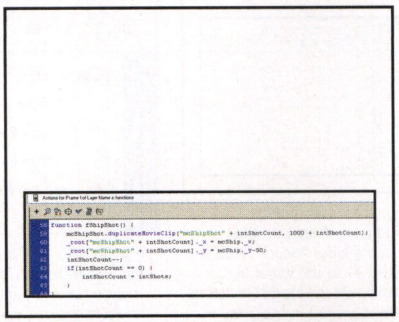
10 Add another function called fStepDown() that moves the mcAlienContainer downwards by the value in intMoveDown every time they reach either side of the screen. Negate the value in intStep to make the aliens move in the opposite direction next time the function fStep() is called. Decrease the value in tAlienSpeed, making the aliens gradually move faster and faster every time they turn.



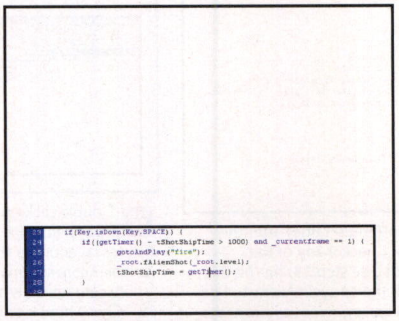
11 Create a new movie clip (Ctrl+F8) and name it mcController. Exit the clip and go back to the main timeline. Drag the movie clip out on the stage. Select it and press F9 to bring up the Actions panel.



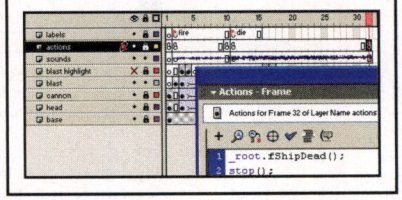
12 Add an onClipEvent(load)and set the variables tThen and tThen2 to the current time. Add an onClipEvent(enterFrame). Check if the current time minus tThen is greater than the value in _root.tAlienSpeed. If so, call the function fStepDown or fStep, depending on the state of nStepDown. This will make the aliens step sideways with an interval of _root.tAlienSpeed and step downwards when they reach the end of either side of the stage. Reset tThen to the current time again.



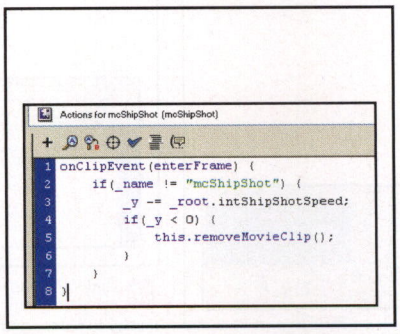
13 Go back to the main timeline and select the first frame on layer a: functions. Add a function called fShipShot(). Duplicate mcShipShot into a new movie clip with the name mcShipShot, plus the variable intShotCount. This will create movie clips which alternate their names between mcShipShot1 and mcShipShot2 every time the ship is shooting, permitting more than one duplicate on screen at the same time. Set its position to line up with the ship.



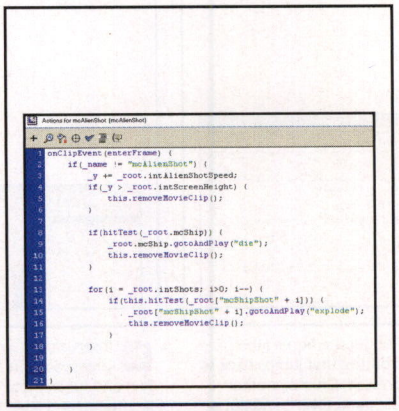
14 Select mcShip and press F9 to bring up the Actions panel. Add an onClipEvent(load) and set the variable tShotShipTime to the current time. In the onClipEvent(enterFrame) add a check for the Space key and go to frame "fire" if one second (1000ms) has passed since the last time Space was pressed. Add a call to a function called _root.fAlienShot() that will let the aliens shoot back when they're being shot at.



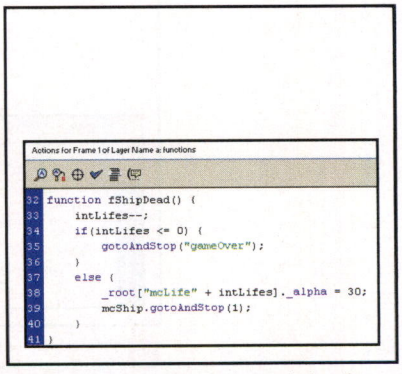
15 Double click on the mcShip and add a call to _root.fShipShot() on the Actions layer under the label "fire". This will time the ship's fire animation and sound with the function that triggers the mcShipShot movie clip. Also add a call to _root.fShipDead() on the last frame.



16 Go back to the main timeline and select mcShipShot. Press F9 to bring up the Actions panel. Add an onClipEvent(enterFrame). Check that the name of this movie clip is not mcShipShot as the code here only should affect duplicates of mcShipShot and not the original movie clip. Move it upwards on the screen by decreasing its y position with _root.intShipShotSpeed. Remove the movie clip if it goes off stage.



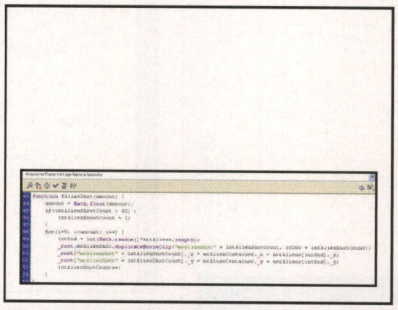
17 Select mcAlienShot. Add similar code here as in mcShipShot but let it move downwards. Check if it hits mcShip and tell mcShip to die if it does. Also, check if it hits one of the duplicates of mcShipShot and tell it to explode if it does.



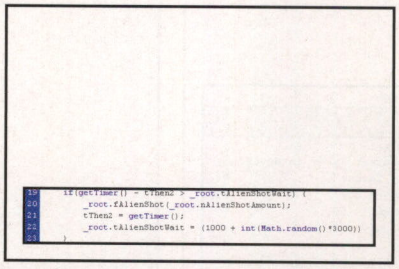
18 Go back to the main timeline and select the first frame on a: functions. Add a function called fShipDead(). Decrease the variable intLifes and go to a frame labelled "gameOver" if all lives are gone. If there are lives left, fade out one of the duplicate mcLifes by changing its alpha value (see step 17 in part 1). Reset mcShip to its first frame. ›

Speeding up sound effects and music

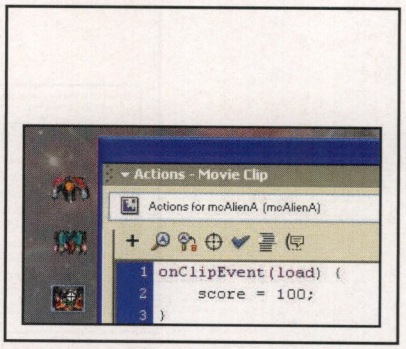
If you end up using a lot of sounds with MP3 compression you may find that it take an awful lot of time to publish your movie. Changing the MP3 compression to "fast" in the publish settings (File>Publish Settings) is a good way to temporarily override the more time-consuming "best" setting on your sound clips without having to restore all your individual settings.



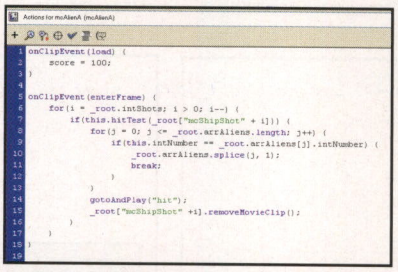
19 Add another function called fAlienShot() that takes the parameter amount. Randomly pick one of the aliens from the array arrAliens. Position a duplicate of mcAlienShot at the picked alien's position. Repeat this for as many times as the value in variable amount, creating new duplicates of mcAlienShot every time.



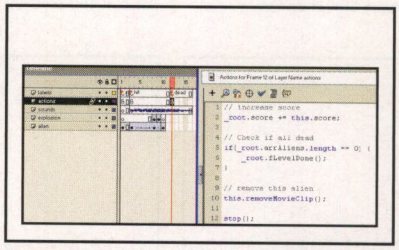
20 Select the movie clip mcController created in step 11. Add a new if statement that triggers the _root.fAlienShot() at intervals of _root.tAlienShotWait. Send in the parameter _root.nAlienShotAmount to tell it how many alien shots to generate. This gets increased by a third every time fStepDown() runs, making one more alien shot every third time the aliens steps downwards on the stage (see step 12).



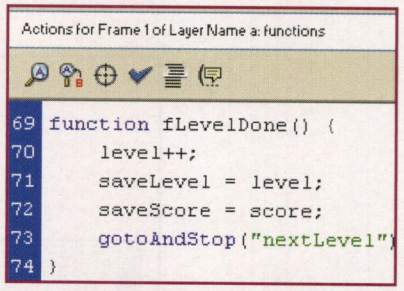
21 Double click on the movie clip mcAlienContainer on the stage. Select the movie clip mcAlienA that resides inside it and press F9 to bring up the Actions panel. Add an onClipEvent(load) and set the variable score to 100.



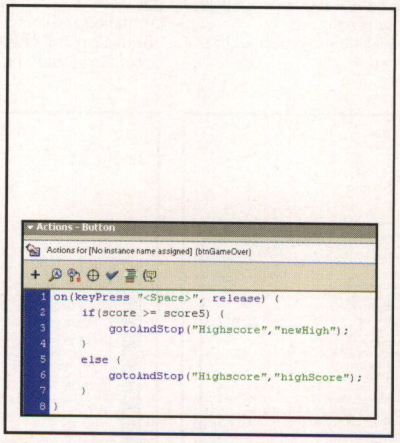
22 Add an onClipEvent(enterFrame) after the onClipEvent(load). Check if any of the duplicates of mcShipShot (see step 13) has hit this alien. If so, loop through the array arrAliens to find this alien's position in the array, remove it from the array and break out of the loop. This will prevent the dead alien from being triggered by fAlienShot() (see step 19). Tell the alien to go to frame "hit" and remove the duplicate of mcShipShot that has hit it.



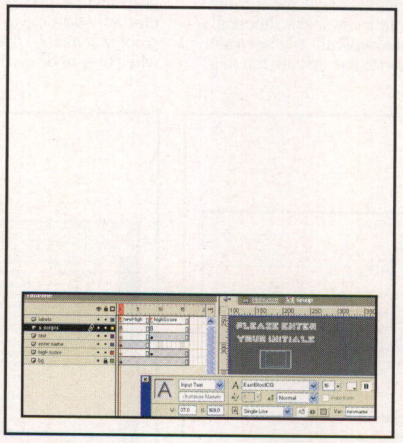
23 Double click on the movie clip mcAlienA to go into its timeline. Add a key frame on the Actions layer under the frame label "dead". Add the variable score set up in step 21 to the variable _root.score which is mirrored on the stage (see step 18). Check how many aliens are left in the array arrAliens. If none are left, call the function _root.fLevelDone(). Remove the movie clip. Repeat the process from step 21 with mcAlienB and mcAlienC but give them different score values.



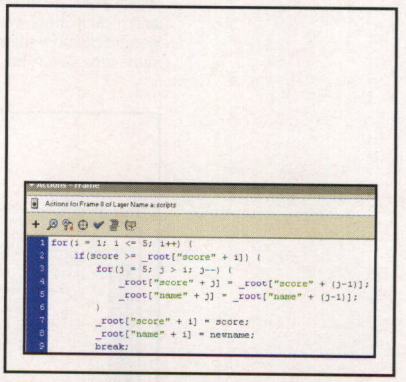
24 Go back to the main timeline and select the first frame on the a: functions layer. Add the function fLevelDone(). Increase the variable level and store the variables score and level in a pair of new variables. Insert a gotoAndStop frame label called "nextLevel". Create two new frame labels on the main timeline called "nextLevel" and "gameOver".



25 Create a keyframe under the frame label "nextLevel". Put a button that jumps back to the label "start" when pressed. Create another keyframe under the label "gameOver" and put a button here with code to check if the current score is higher than the lowest score in the high score list. Go to a frame labelled "newHigh" in a new scene called Highscore if there's a new high score. Otherwise go to a label called "highScore".



26 Insert a new scene (Insert>Scene) and name it Highscore. Create layers for labels, a: scripts, text and for any graphics you want. Create two keyframes and label them "newHigh" and "highScore". Under the frame labelled "newHigh" put a stop(). On the a: scripts layer add an input text field with the Var field set to newname. Now add a button that goes to the frame labelled highScore when pressed.



27 Under the frame labelled "highScore" put five dynamic text fields in two columns named name1 to name5 and score1 to score5. Add a script to the frame in the a: script layer that checks which position in the list the new score has. Move down the other high scores from that position, leaving the bottom one to go off the list and add the new one to its position in the list. Add a "play again" button that jumps to label "start" in the scene where the game is. Finally, add a new scene called "intro" that comes before the game starts. Initialise the variables score1 – score5 and name1 – name6 here. ca•s

SAVE £24

SUBSCRIBE TO *COMPUTER ARTS SPECIAL* AND GET 13 ISSUES FOR THE PRICE OF 9

UK: 0870 444 8455

OVERSEAS: +44 (0) 870 444 8455

OR ORDER ONLINE: www.computerarts.co.uk/subscribe

quoting order code: COSP40 Lines open 24 hours a day

This is your chance to subscribe to *Computer Arts Special* for just £54. You'll get every issue delivered to your address, and more importantly, you'll save £24 on the deal. Phone our hotline today!

Computer Arts Specials are certainly hot property – you'll find all the best techniques and tools for a variety of digital art disciplines:

• 3D graphics • Flash • Dreamweaver •
• Digital video • Web design • Photoshop •

Each issue comes complete with a dual-format CD.
Simply call our hotline, email us, or fill in the form on the right.
(A photocopy will do)

Make even greater savings by subscribing to *Computer Arts* too!

Turn to page 91 for details.

☑YES! I want to subscribe to arts special

Cut out or photocopy and return this coupon

Title _____ Initials _____ Surname _____

Address _____

_____ Postcode/Zip _____ Country _____

Daytime telephone number _____

Email address _____

13 Special issues

UK (cheque/credit card) ☐ £54 save £24
Europe (including Eire)* ☐ £73.53
Rest of the world* ☐ £86.53 *AIRMAIL

Please choose your method of payment 1, 2 or 3

1 ☐ **Cheque** (payable to Future Publishing Ltd. Sterling cheques drawn on a UK account.)

2 ☐ **Visa** or ☐ **MasterCard**

3 ☐ **Switch** Issue No/Valid date _____

Card No _____ Expires _____

Signature _____ Date _____

UK readers return this coupon (no stamp needed) to: *Computer Arts* Magazine, Subscriptions, Future Publishing Ltd, FREEPOST BS4900, Somerton, Somerset, TA11 6BR.

Overseas readers return (postage payable) to: *Computer Arts* Magazine, Subscriptions, Future Publishing Ltd, Cary Court, Somerton, Somerset, UK TA11 6TB.

☐ Please tick this box if you would prefer not to receive information on other offers.

Order code: COSP40 **Offer ends 15.01.03**

Get behind the camera...
Climb into the Director's chair with Toon Boom Studio V2! We've just added a whole new dimension to 2D animation production that lets you dive right into the action and take total cinematic control of your creations.
www.toonboomstudio.com

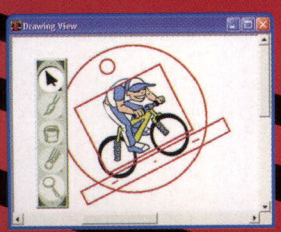
action!

TOON BOOM STUDIO V2

2D ANIMATION SOFTWARE

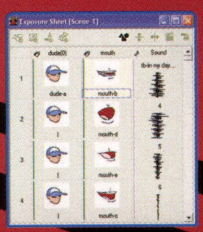
Animate

Breath life into even your wackiest designs and characters with specialized animation tools that offer endless expression.



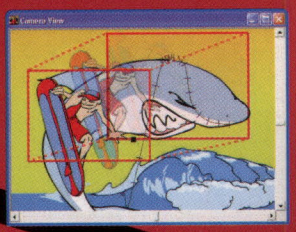
Lip-sync

Precision sound editing tools and a lip-sync generator let you match action to sound effortlessly and automap mouth drawings to voice tracks in any language.



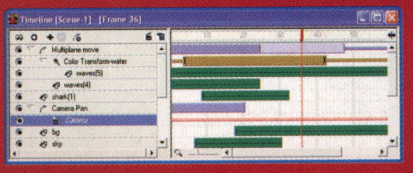
Stage

The multiplane camera lets you move among your characters on a 3D stage to choreograph and integrate stunning cinematic effects.



Edit

The timeline window gives you precise control over the timing of each element and multi-dimensional tweening effects while visualizing the progress of an entire scene.



Reuse

The visual library lets you reuse or share your work to build collaborative projects.

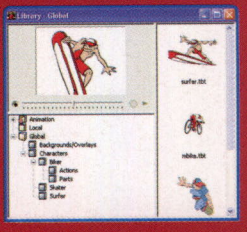


Illustration: Rod Hunt [e] rod @rodhunt.com [w] www.rodhunt.com

Bring your game characters to life

We show you how to animate characters in *Flash*, and we demonstrate how to introduce more texture to your *Flash* layouts using bitmaps and masks...

Flash has had a huge impact on the animation industry. Its easy-to-use interface, low file size output and quick tweening methods make it an essential tool for Web and broadcast animators. In the first part of this tutorial, we take a static character illustration from a *FreeHand* file and show you how to bring it to life with a simple walking animation in *Flash*. We cover the modularisation of different animation elements and complex shape tweening, both of which are effective techniques in the creation of believable, compelling animation. We also explore some basic ActionScript techniques that allow you to manipulate different sequences so they don't become too repetitive.

The last part of the tutorial is a step-by-step guide to creating a *Flash* composition which seamlessly combines bitmaps and vectors. This approach enables you to produce designs with far more depth and texture than working with simple vector shapes.

The tutorial is intended as a basic introduction to some of the techniques used in *Flash* animation, although advanced users might find a few cheeky tricks in there too! The creative subject of both guides is an interactive game called *Make Or Break*, designed for our client The Foyer Federation.

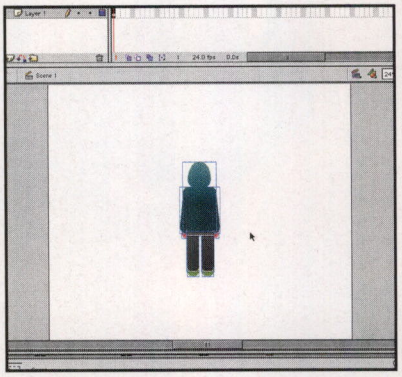
Expertise provided by Tom Evans
[e] tom.evans@mook.co.uk and Tom Adams
[e] tom.adams@mook.co.uk at Mook.
Visit the Website at [w] www.mook.co.uk

Files on disc
The files needed to support this tutorial can be found in the Tutorial\gm_mook folder of the cover CD.

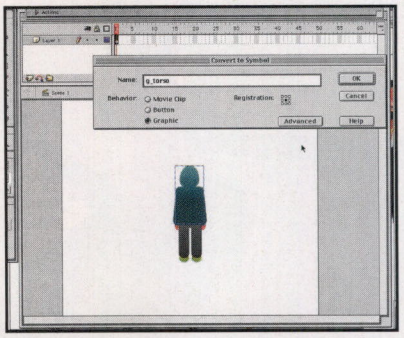
Part 1: Getting started

Importing your object and converting it to symbols…

1 The object we're going to animate is a small variant of the larger one – it will only be around 75 pixels high. This means we can reduce the illustrative detail and simplify the animation.



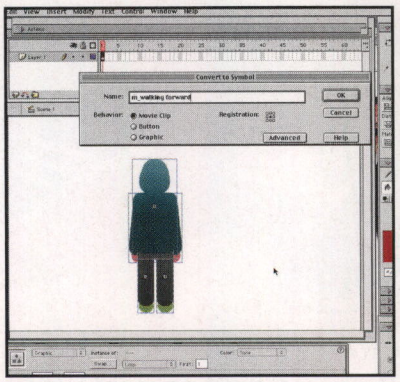
2 The object is copied from *FreeHand*, pasted into *Flash* and broken down into separate elements. This also works in *Illustrator*.



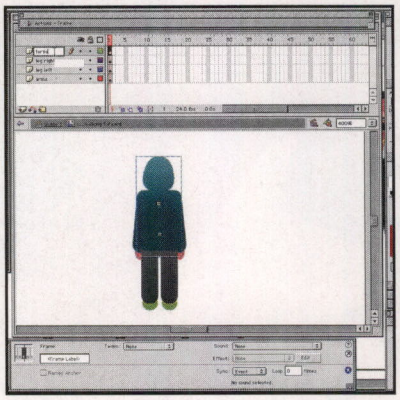
3 All the elements need to be converted to symbols (key F8) before you can start animating. The legs are the same, so you just need one symbol for those. The static elements will be graphics and the rest will be in movie clips.

Part 2: Initial movement

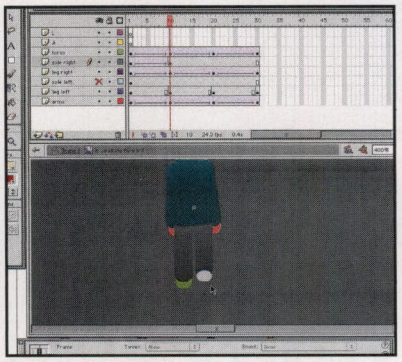
Make a movie clip and get your character moving…



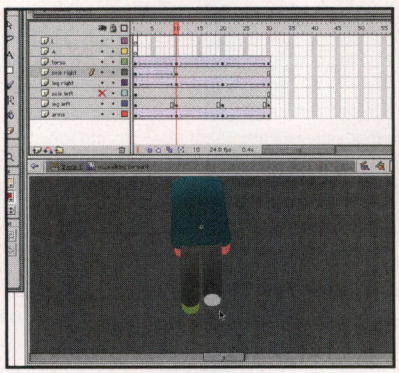
1 After you've converted all of the elements to symbols, they need to be converted into one movie clip. You might find it helpful to use _g as a prefix to the graphics and _m for the movie clips so they group nicely in the Library.



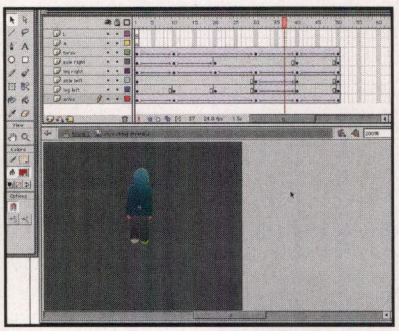
2 Next, you need to create separate layers within the movie clip so you can animate the different elements. You can only have one symbol per layer if you're going to use tweens.



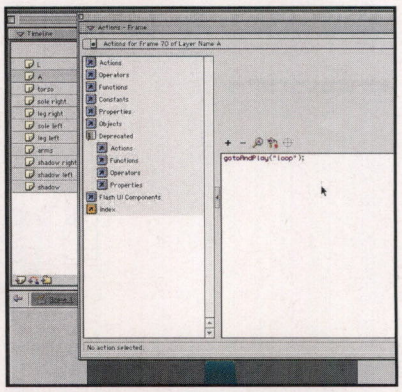
3 Now the really tricky bit begins: getting the movement right. The torso just rotates a bit and the same goes for the arms. They need to be fractionally out of sync to make it look natural. The legs are scaled up and down to suggest movement. You can only use motion tweens for this.



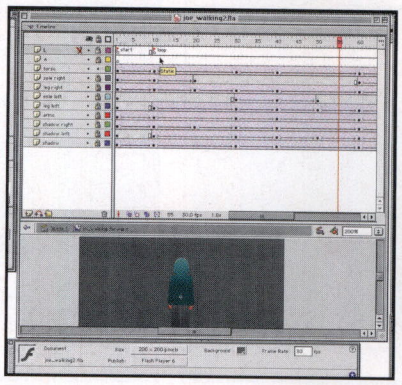
4 To create the illusion of the leg bending, you can use a white sole that fades in as the leg rises. This trick works particularly well at this scale and level of detail.



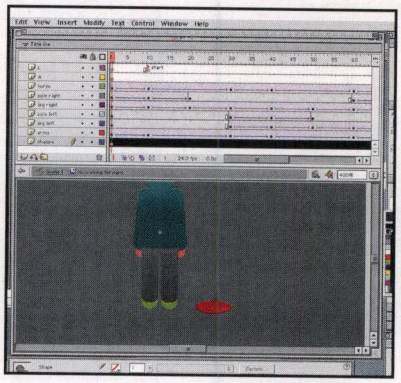
5 The next task is to complete the movements for both sides so you can loop the whole movie clip. Because the movement is slightly different when the character starts to walk compared to when he's actually walking, you need to create an 'overlap' that's almost the same as the start.



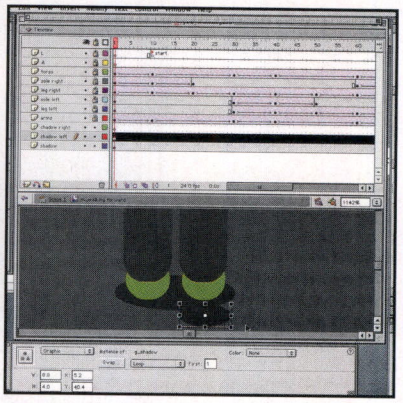
6 At the end of the timeline, create a keyframe (key F6) on the A (actions) layer that makes the movie clip go to 'loop'.



7 Jumping to the 'loop' frame bypasses the specific opening walk animation once the character is on the move. This helps to create a smooth walking movement.



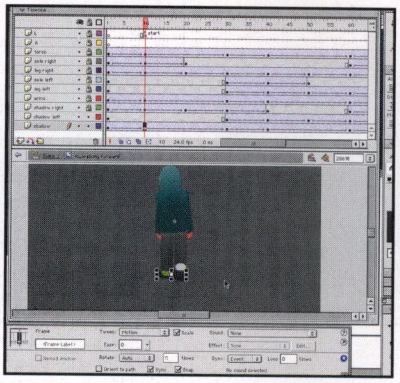
8 The next step is to create a drop shadow, which increases the 3D effect. Just draw an oval that's about the same width as the character, give it an appropriate colour and create a symbol.



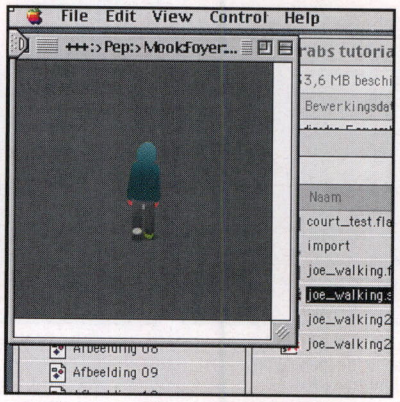
9 To create the impression of the feet lifting off the ground, we need to create shadows for both legs in two further layers.

Part 3: Motion and shape tweening

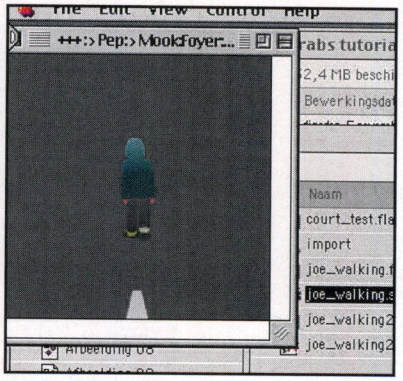
Get your character walking, looking around and behaving realistically…



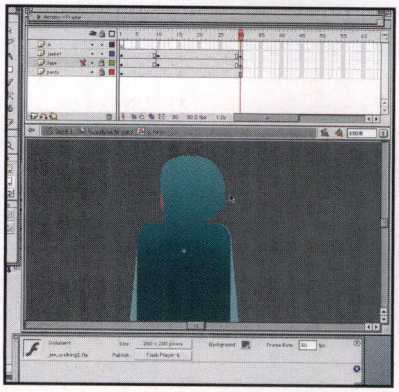
1 You now have to motion tween the leg shadows so it looks like the feet lift from the ground by just moving them up and down.



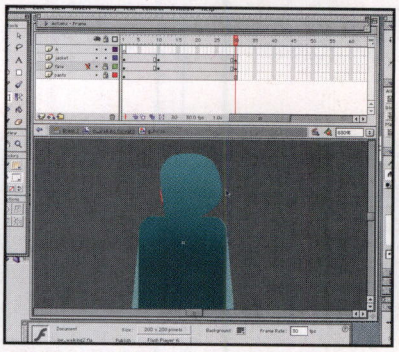
2 The motion tweening on the legs is now finished and our character can walk!



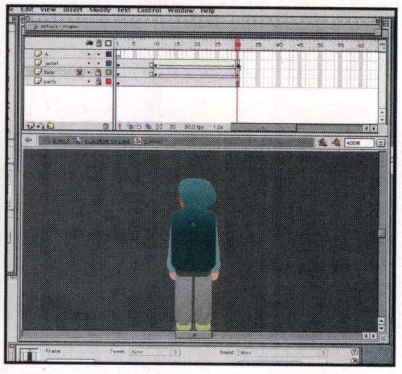
3 For extra realism, it'd be good to make him look around a bit and occasionally stick his hands into his pockets.



4 To do this, go into the torso movie clip and create separate layers for the character's jacket, face and bottom. You don't yet have a graphic for his face, so create another oval with a pink gradient.

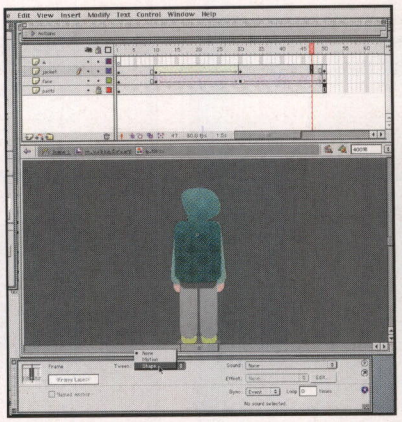


5 You can make your character look sideways simply by simultaneously changing the shape of his hood and making the face appear. Just use the arrow to pull the outer line of the shape. You should try not to delete or add points as this can spoil the shape tween.

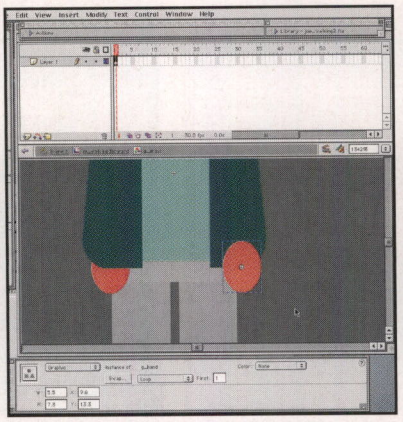


6 Because you're changing the shape of the hood, you need to create a shape tween instead of a motion tween. This is quite easy with the hood, but the back of a head would be a little more complicated. ›

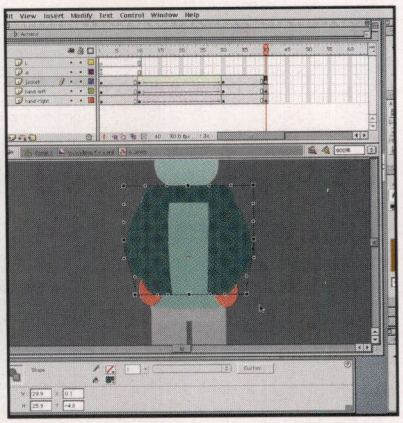
Part 3: Motion and shape tweening continued...



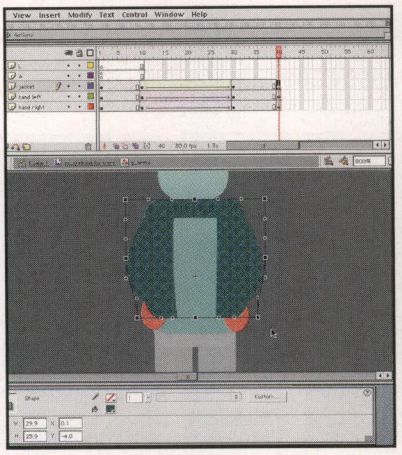
7 To create a smooth movement, the elements have to return to their original shape and position at the end of the timeline. To do this, copy the first frame, paste it at the end and create another shape tween.



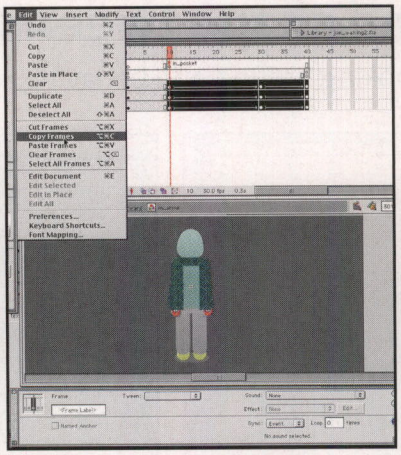
8 The next step is to make your character move his hands into his pockets. Use the symbols you've created for the face and the hands and place them on separate layers. The sleeves/arms should go on a third layer on top of the hands.



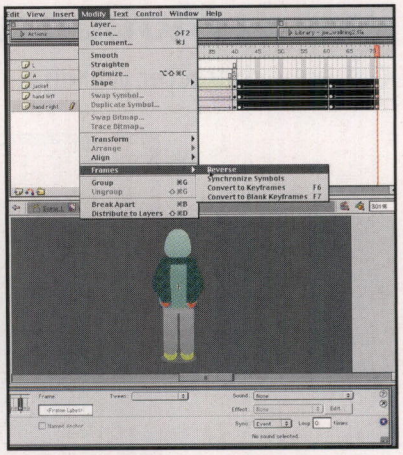
9 The hands move up and disappear behind the jacket with a motion tween. You need to use a shape tween to suggest him pulling up his arms, bending his elbows and finally putting his hands into his pockets.



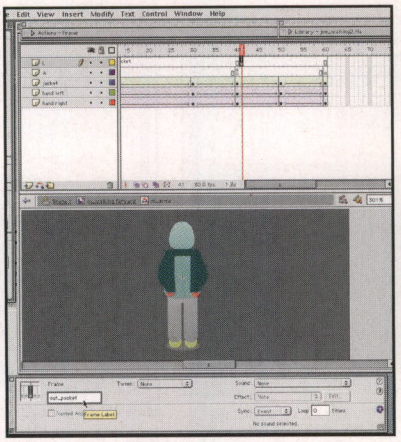
10 In *Flash MX* you can use the Envelope or Distort tool to change the shape. These functions sit under the transform tool.



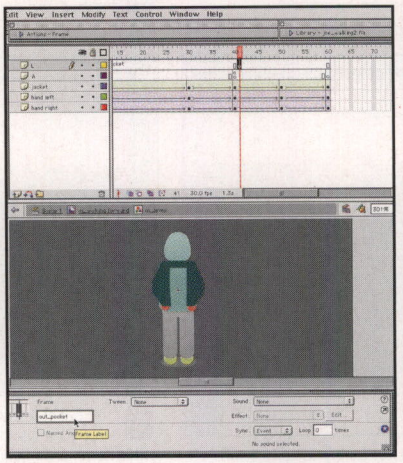
11 Another way of moving your character's arms back out of the pockets is by copying the frames with the tweens.



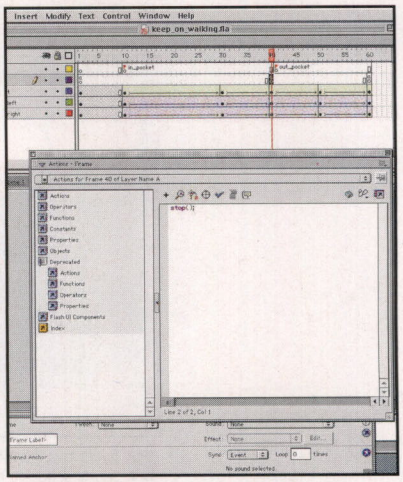
12 Now paste the frames behind and reverse frames. You can find this function under the Modify>Frames menu.



13 To make sure this movement doesn't become too repetitive, add a keyframe with a "Stop" action and a frame label at the start so you can trigger the movement externally.



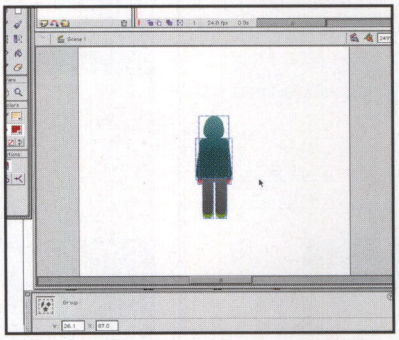
14 Label the frames "in_pocket" when the hands go into the pockets and "out_pocket" when the hands come out.



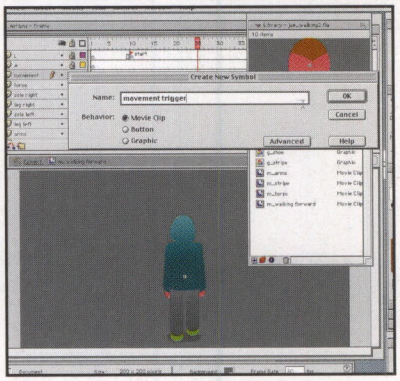
15 Make sure there's a "Stop" action at the end of both of these movements.

Part 4: Triggering actions

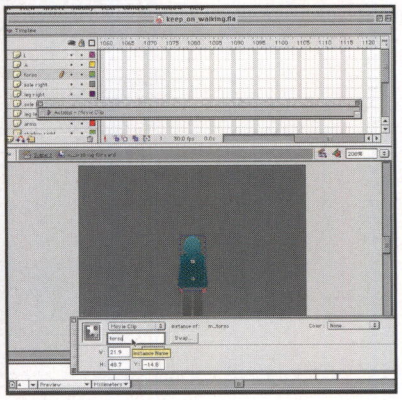
Adding keyframes to the timeline to keep your character moving…



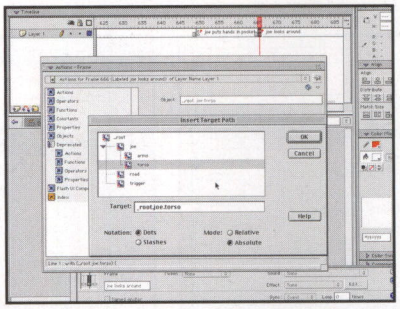
1 Your next task it to create a movie clip which triggers the 'looking around' and the 'hands in pockets' actions. This could be done with advanced ActionScripting, but we kept it simple by using a timeline with keyframes.



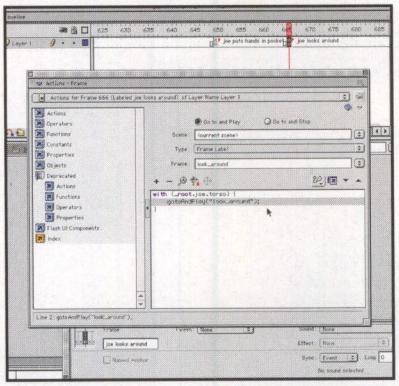
2 Start by adding a few keyframes along the timeline – this one is about 1200 frames long. Give them a label to keep track of what they're for.



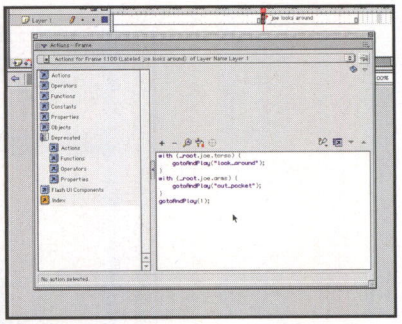
3 Before targeting movie clips you have to give them an instance name.



4 Select the keyframe and open the ActionScript window. Find the action "with" and use the little target symbol to find the torso movie clip. Because the torso sits in another movie clip, you have to name that one as well. In our example, this one's named "Joe".



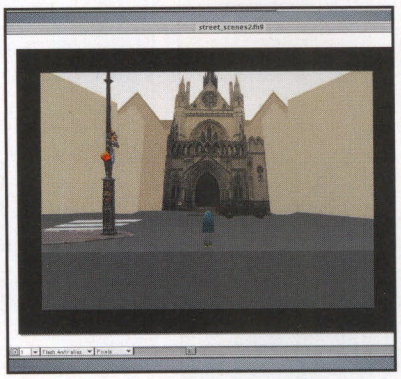
5 Now add "gotoAndPlay" and type the frame label "look_around".



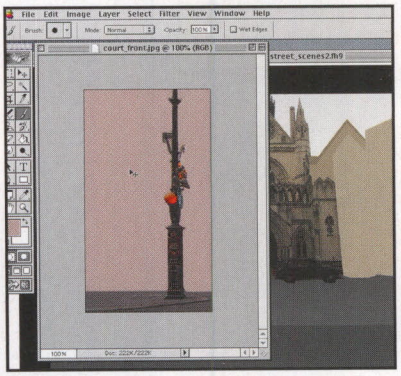
6 You can play with timing and target more than one movie clip from the same keyframe. This is the last keyframe of the movement trigger movie clips, and the gotoAndPlay frame 1 instruction ensures Joe keeps walking.

Part 5: Combining vectors and bitmaps

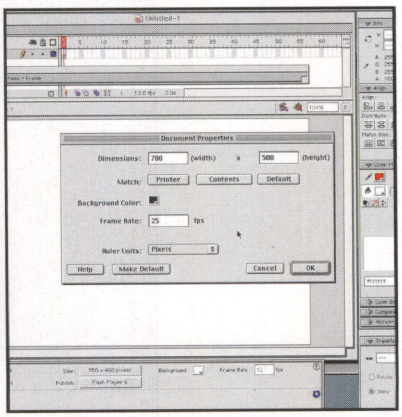
Integrating your character with its environment…



1 This is the layout as it is in *FreeHand*. Everything you can see is vector graphics, except for the front of the building and the street lamp. These are both bitmaps and need to be scaled and converted to jpegs. Jpeg is usually the best format for photography.

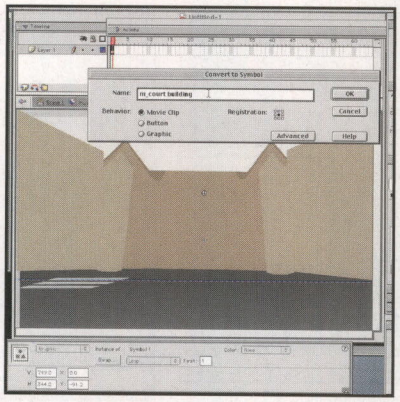


2 You also reduce file size by filling the areas that are going to be masked with a solid colour. Pick a colour that blends nicely into the background, in this case a warm grey. Once the bitmaps have been created in *Photoshop*, you can start working in *Flash*.

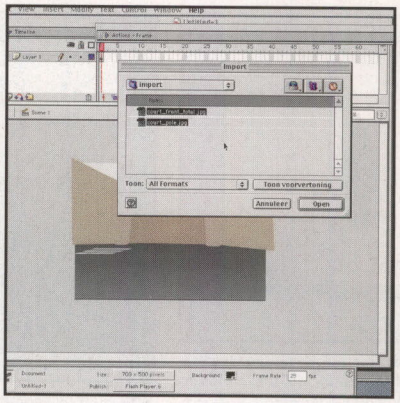


3 Create a new file in *Flash* and define the size, pick a background colour and a framerate. To get smooth animation it's best to use 25 or 30 frames per second.

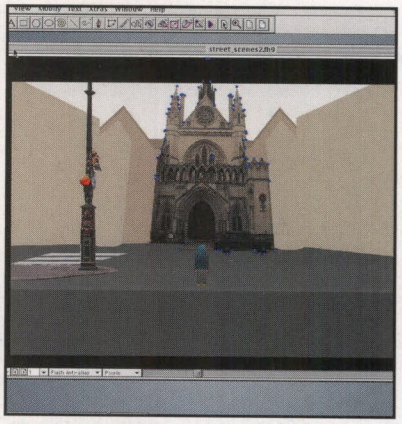
Part 5: Combining vectors and bitmaps continued...



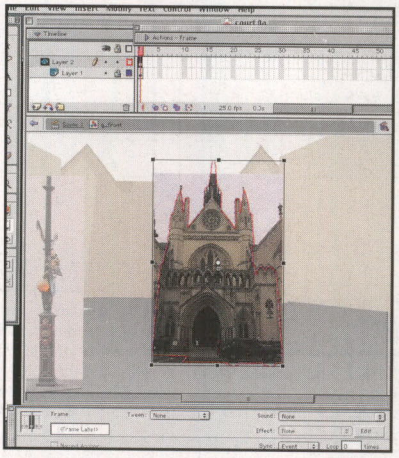
4 Copy and paste the vector graphics into the *Flash* file and convert the separate elements to symbols. Elements that are going to be animated should become movie clips; the rest can be graphics.



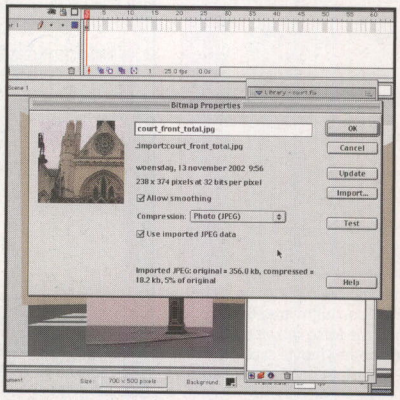
5 When all of the vectors have been imported, the next step is to import the bitmaps. Once again, you need to make the bitmaps into symbols so key F8 to do this.



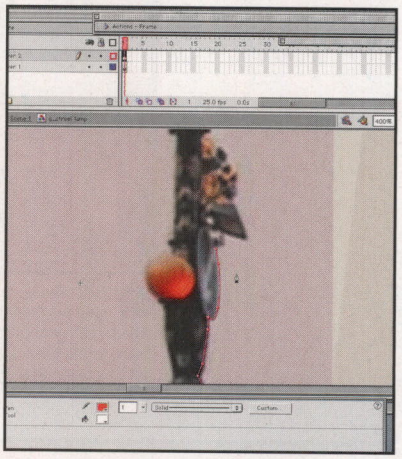
6 Now copy the mask you created in *FreeHand* and paste it into a separate layer in the bitmap graphic in *Flash*.



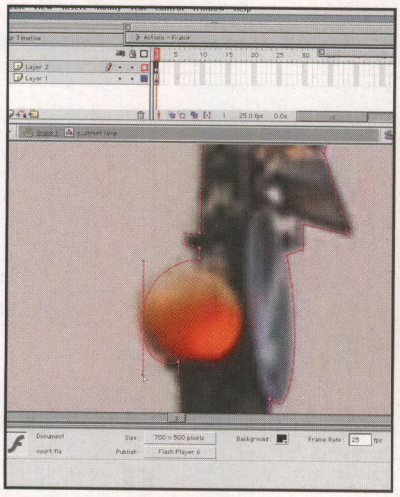
7 Position it on top of the bitmap and create a Mask layer by holding the right mouse button on the layer and selecting "mask".



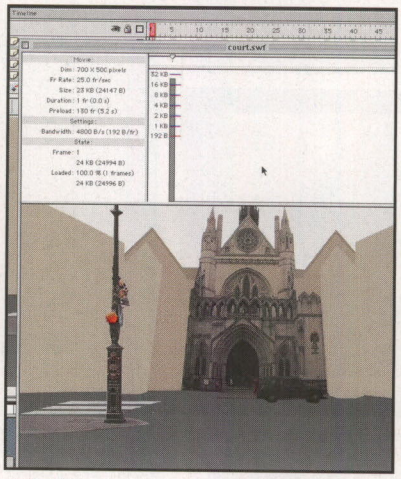
8 You can check the properties of the bitmap by double-clicking on the Bitmap icon in the Library. Since we've already applied jpeg compression from *Photoshop*, we use "imported jpeg data", but you can also use the jpeg compression in *Flash* itself.



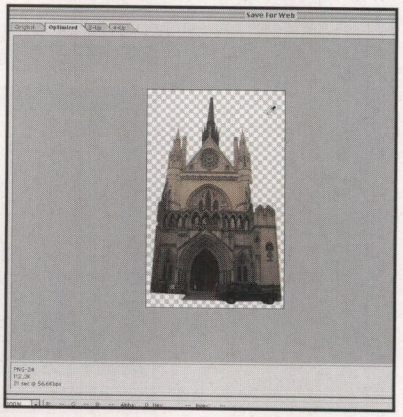
9 Another way of creating a mask is to draw it with the Pencil tool in *Flash*. Remember, the fewer points you use, the smaller the file size will be.



10 Use the white arrow (sub-selection tool) to adjust points and curves.



11 All the elements are now in place and you can check the file size by testing the movie (using Command-Return). Our file is 23KB.



12 Alternatively, you can use the PNG format to create the same effect we've been talking about. This works with an anti-alias alpha and avoids the jagged edges of gifs. Once you've imported it into *Flash*, you can apply jpeg compression in the Bitmap Properties window. ca-s

THE MAGAZINE WHOSE TIME HAS COME...

...A magazine dedicated to style, connectivity & convergence.
Technology for life

INSIDE THE FIRST EDITION
SYNERGY & STYLE Enjoy seamless technologies for smarter living **SURROUND SOUND REVOLUTION** One speaker panel to rule them all? **HOUSE 2010** Fastforward to the future home **FREEDOM TO ROAM** Get broadband internet on the move **VIDEOTAPE IS DEAD!** Every new recordable technology tested

Get connected. Get digitalhome. **On sale Dec 19**

BRAND NEW £3.50

Order your copy direct (UK only):
TEL **0870 444 8657**
FAX **01458 271146**
ONLINE **www.digitalhomemag.com/issue**

future publishing
Media with passion

Kerb

"Many 8-bit games can be recreated in *Flash*, with better graphics and sound," says Kerb's Jim McNiven. We ask him how to make *Flash* games, and why his Website became one...

Founded in 1996, Brighton-based Kerb is well known within the world of new media design. High-profile clients such as BT, Disney and the BBC have been won over by the company's innovative and creative stance. While initially concentrating on illustration and animation, Kerb has now ventured into television, music videos and *Flash* games.

Perhaps logically then, the latest incarnation of Kerb's Website is a *Flash* game. As Creative Director, Jim McNiven, explains: "We had a nightmare thinking of how to present the services we offer. It was hard to generate a theme to casually explain that we create Websites, games, viral campaigns, music videos and our own television series."

Website exploration

With a team keen on gaming, they decided to create an imaginary version of the Kerb offices, and let users explore via an 'old skool' platform game. Coding groundwork had been done while creating the *Travel Sick* game for Bravo, so most additional work involved integrating a database for regular news releases, to keep the site fresh.

Responses to the Kerb site have been positive, and some clients had to be dissuaded from wanting something identical themselves. "Unsurprisingly, feedback from designers has been mixed," says Jim. "Some appreciate the extravagance, style and tight coding, but others bitched about it being complicated and too time-consuming to navigate, claiming it would scare potential clients."

As Jim says, the latter point is moot — Kerb targets Web-savvy markets rather than SMEs (Small-to-Medium Enterprises) wanting 'generic' Website design. "To that end, representing our work via a game was probably a no-brainer."

The site contains many quirky features, including a *Blade Runner*-inspired top floor, and clients being tortured on the ground floor, according to Jim, "as visual representation of the pain they go through with us."

Long-time gamers will also detect a hint of nostalgia within. "The game is retro, based on a C64 game I used to love called *Pyjamarama*," explains Jim. Despite being an ex-member of C64 and Amiga demo crews, Kerb's use of such game ideas is

1. The ground floor of the first level of *Panik In Chocoland*. In this arcade-style platform game, you have to collect bombs to progress through each level. The game was illustrated in *Flash 4* and programmed in *Flash 5*.

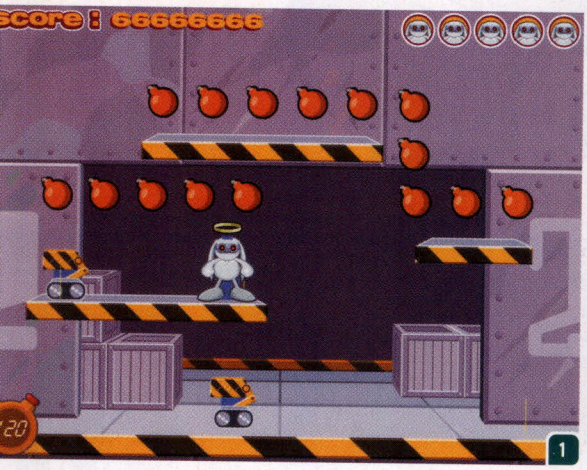
2. Another image from *Panik In Chocoland*, which shows the top of the first level. Here, you can see the escape door that appears once you've collected all the bombs; it also features the head of the almighty Bob Dobbs.

3. *Travel Sick* is a platform adventure game consisting of three levels of increasing difficulty. In level one you can give this watermelon-obsessive the object of his desire and swipe his pliers when he's getting 'stuck in'.

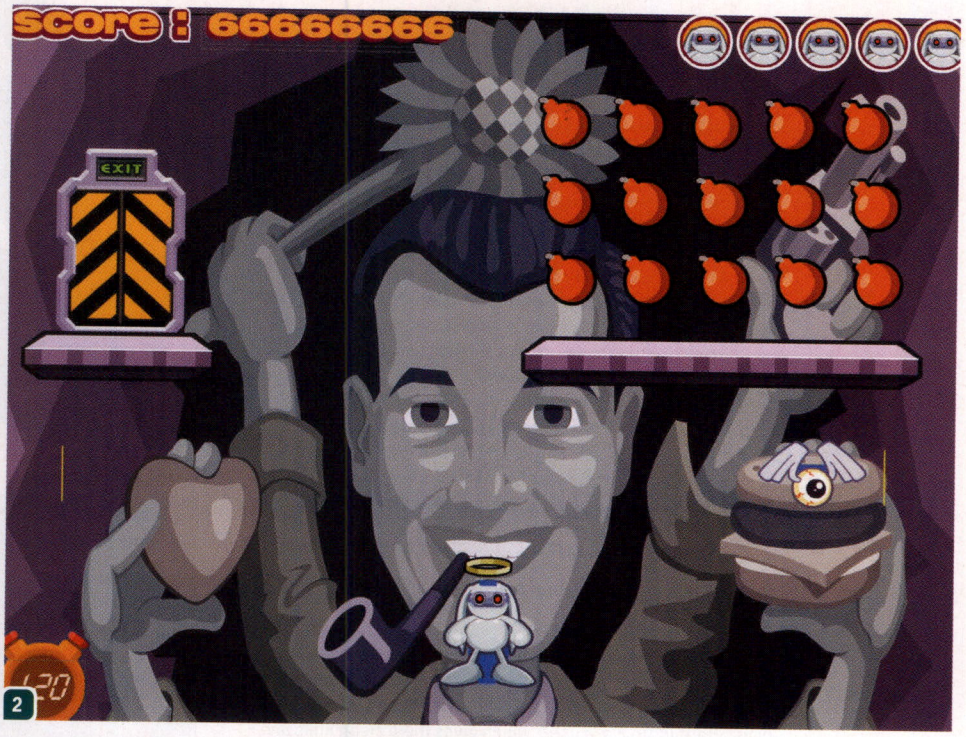
4. *Travel Sick's* gameplay relies on a mix of lateral thinking, problem solving and quick reactions to negotiate the terrain. In this image we see *Loaded* magazine and TV's very own Grub Smith mere seconds away from meeting an icy death.

5. This is the second level of *Travel Sick*. The mission is to give the cat a Viking funeral. Depicted is the 'victim', but first you have to work out how to poison it. Like the bulk of Kerb's games, *Travel Sick* was built in *Flash 5*.

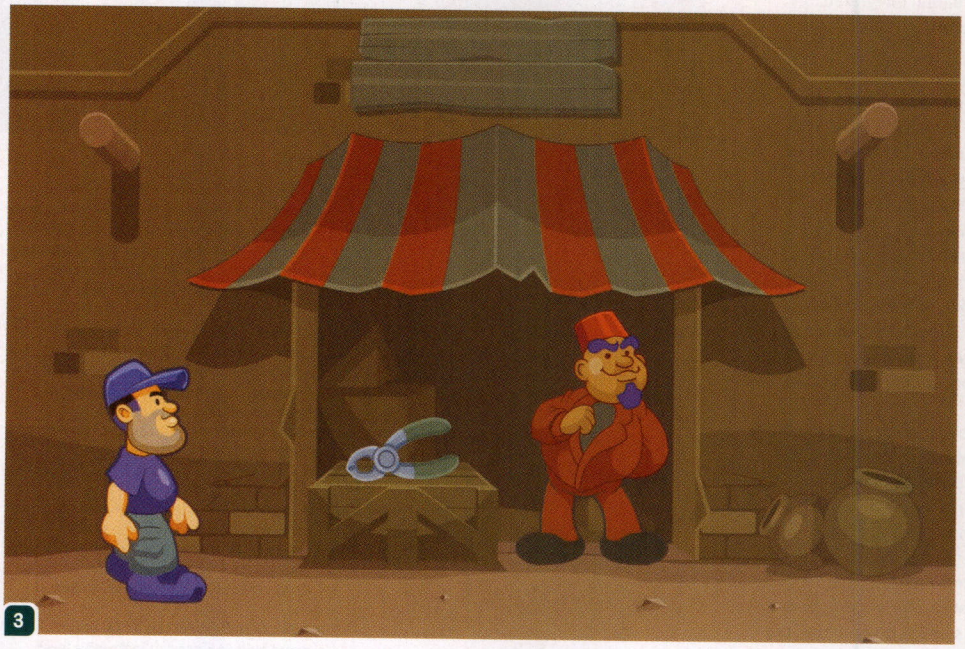
6. Classic 8-bit computer games, such as *Bounder* and *Pyjamarama*, provide inspiration for many of Kerb's *Flash* games. We're hankering for an online 'version' of *Wizball* or *Miner 2049er* from them next time round.


score : 66666666

1


score : 66666666
EXIT

2



3

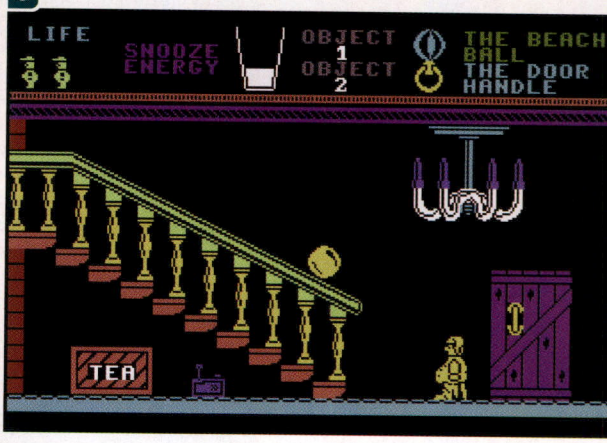
YOUR SCORE
490
JUMP
40

6

4

VET

5


LIFE
SNOOZE
ENERGY
OBJECT
1
OBJECT
2
THE BEACH
BALL
THE DOOR
HANDLE
TEA

not down to rose-tinted spectacles. "Many 8-bit games had great gameplay," says Jim. "And they can mostly be recreated in *Flash*, with better graphics and sound."

Jim reckons *Flash* is an excellent environment for developing games, setting a level playing field, which mirrors the competitive environment of days passed. "It's versatile and cross-platform," says Jim. "Everyone has to push it as far as possible until a new version appears. We're constantly discovering amazing new stuff by people like Titoonic and wondering how things were achieved."

So what makes a great *Flash* game? "It shouldn't need instructions and must have a perfectly weighted learning curve," claims Jim. "A sense of humour is helpful, and while you might get frustrated with your ability, that shouldn't be the case with the game itself. Truly great games always give you that 'one more go' feeling. That's why I love Nintendo. Its designers know how to grab you and turn up the heat to keep you hooked, but not so much that you switch off." But he admits that creating online games is rather different to authoring for consoles or PCs: "Budgets for *Flash* games are often non-existent. Therefore expectations are lower, although we hope people playing our games get good value with regards to investing their time."

Programming tips

As for making the games themselves, programming comes first. This means programmers can work with dummy graphics until satisfied with frameworks. Meanwhile, illustrators design in-game graphics in *Flash* itself, which gives them clean lines and reduces file sizes. Finished graphics are then integrated, programming polished, and, finally, sound and high-score tables added.

Common problems include game speed and parallax scrolling. "*Flash* is fairly processor intensive and too much on-screen action causes things to lag," says Jim. He advises keeping moving objects to a minimum and using flip-screen scrolling. To speed up parallax scrolling, Jim recommends breaking apart objects on each layer: "Instead of scrolling multiple objects, *Flash* only has to deal with one per layer." Further tips include loading levels in the background while players are exploring, and extra content only when specifically requested. "Also nip over to www.flashkit.com if you're a beginner, and study the tutorials," says Jim. "I won't pretend to know much about ActionScript, but from what I can gather, it's useful to have good resource books and a really big book on applied physics!"

While making games can be fun, it can also be profitable. "We mostly create promotional content for brands," explains Jim. "Although we've also built games in-house, which were licensed to content sites for a fixed 'rental' fee."

Undoubtedly, a budget pays dividends with the games themselves. "Bravo's decent budget for *Travel Sick* is a case in point," says Jim. Kerb was able to make something that looked stunning, and was extensively play-tested. Many *Flash* games suffer from a lack of play-testing – this keeps budget lows but makes it harder for companies to create playable games.

Regardless, Kerb continues to develop *Flash* games. "I can't say much about our next project, as it's all 'hush hush'," says Jim. "But I can reveal it's a healthy mix of *Flash*, database stuff and community gaming. Our most recent *Flash* game was for BT Broadband, which involves the BT pigs flying through a pipe (Internet connection), collecting files and being slowed down if they don't get Broadband." Now that's entertainment. ca-s

You can see Kerb's games at **[w]** www.panikgames.com, **[w]** www.bravo.co.uk/travelsick, and **[w]** www.kerb.co.uk. And you can contact them at **[e]** info@kerb.co.uk, **[t]** 0870 013 0040 and **[f]** 01273 682 637

1

2

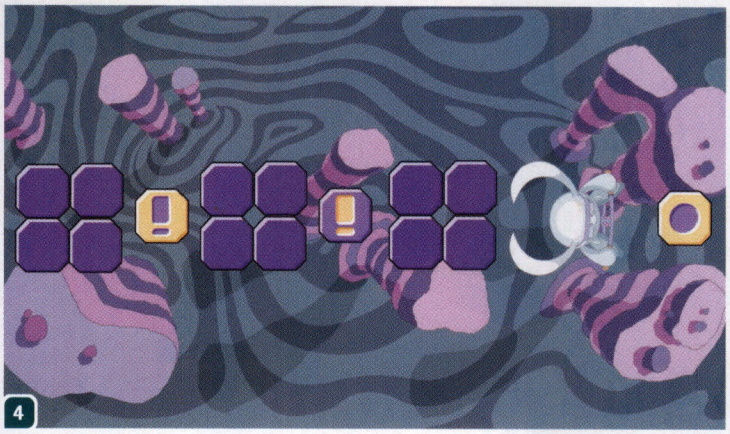
1. The top floor of the Kerb Website. Hovering on the updraft enables you to grab a key-card, which can be used to unlock a door later on in the site. The game/site was programmed in *Flash 5*, and arrays were used to define each room and floor.

2. While the Kerb site is chock-full of useful information, all easily updateable via a back-end CMS, there's also plenty of nonsense, such as this locked room. It harbours a disk that contains screen savers; however, the disk must be inserted into a terminal elsewhere in order to access them.

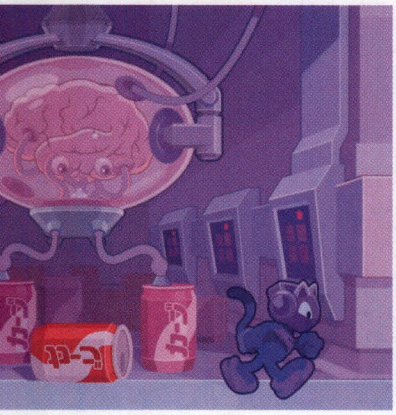
3. Insert a battery into the shrink-ray device and you'll contract to a small enough size to enter the mouse hole. Inside, you can view the psycho vegetables from Kerb's television show, *Hellz Kitchen*.

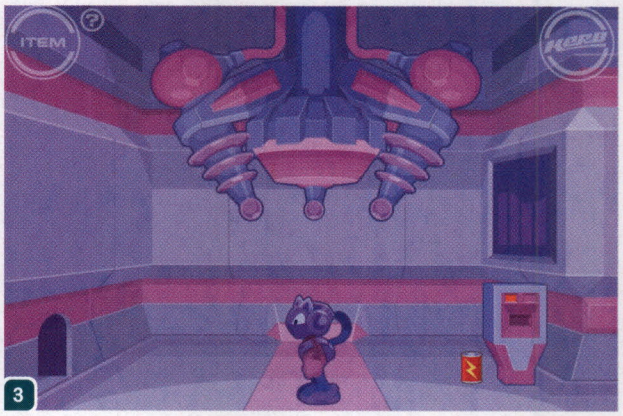
4. *Panik In Pogoworld* is a tile-based game built in *Flash 5*. It includes a level editor, enabling the owner to design and add their own levels. This feature was made possible by the use of multi-dimensional arrays loaded into *Flash* from a text file for each level.

5. This promotional *Flash 4* game is based on the 3D animated Robbie Williams video from a while back. You control Robbie and go shagging around three different levels of scenery, each with parallax scrolling. The icy third level is particularly hard to negotiate, as Kerb whacked up the 'slipperiness' setting to the max.

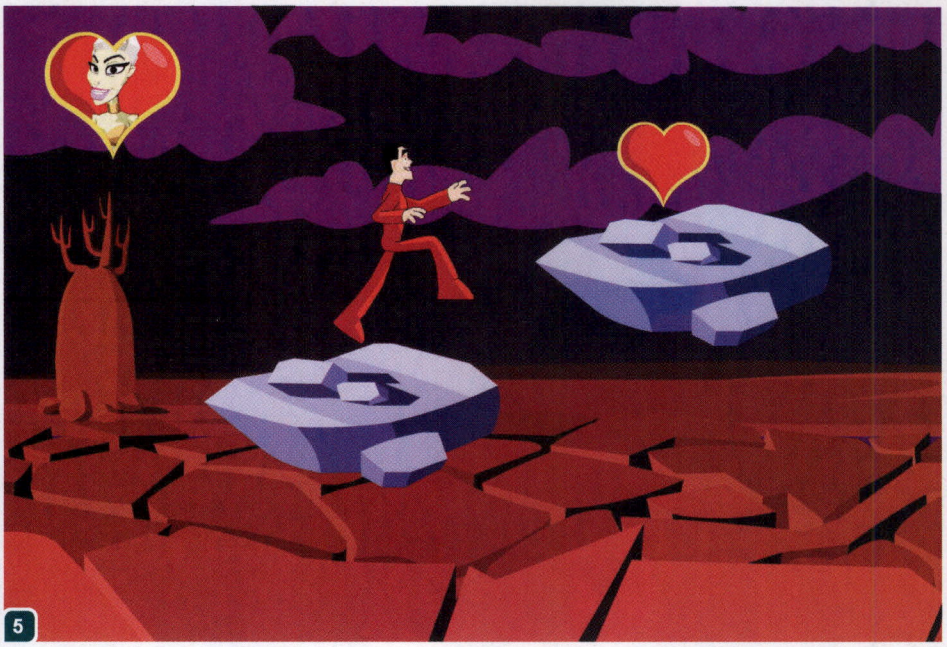


4





3



5

Kerb's flashiest moments

Jim McNiven tells us more about the very best of Kerb's *Flash* games...

Based on classic 8-bit platform game *Pyjamarama*, *Travel Sick* succeeds due to slick graphics and the game's 'feel'. After working out each level's concepts and puzzles, graphics and code were created within *Flash 5*.

Panik In Chocoland also thrives on a mix of playability and polished graphics. "Dave was messing around with the physics for ages and pretty much built the game in his free time," comments Jim. "Then Dantoo came up with a sketch of a boy in a rabbit costume that Dan redrew and animated in a cleaner style. Zeb finished it off by adding some backgrounds." Apparently, when displayed within a cabinet for Melia in February, people were genuinely surprised it was done in *Flash*.

Another 'Panik' game is the third on Jim's list, namely *Panik In Pogoworld*. Inspired by the unique Commodore 64 platform game *Bounder*, Jim cites its main quality as being really addictive. "It's also our first *Flash* game creation with a level editor, so the game goes on infinitely," says Jim. Sensibly, a password system was introduced for higher levels, so you don't have to return to the start each time you play.

Finally, while Kerb's Website isn't strictly a game, it has a large following nonetheless. "We got huge traffic from Spanish gaming sites, resulting in numerous emails from easily-pleased Spanish gamers," comments Jim. He continues: "We hope it showcases nice graphics and tight coding, combined with database skills and a sense of humour."

Unsurprisingly, the site, created between paying projects, rapidly spiralled out of control. "It almost killed Sermad, who did the bulk of the coding," says Jim. "Every time he had it nailed, someone would think up more features!"

As for why it was created, Jim states: "We were trying to create a visual representation of what we do."

100000
80000
60000
40000
20000
0
ACTION
1UP
MovieClip.prototype.shootBullet = function() {
if (_root.bulletOnScreen == false) {
_root.attachMovie("bullets", "liveBullet", 1);
currentBullet = _root["liveBullet"];
currentBullet._x = this._x;
currentBullet._y = this._y;
currentBullet._rotation = this._rotation;
_root.bulletOnScreen = true
macromedia
FLASH
MX

Five essential ActionScript commands for gaming

When it comes to games and interactivity, ActionScript is the prime component for success. We're going to briefly step through five essential ActionScript structures for games programming and compile them all into a simple *Asteroids*-style game.

1 Keyboard control
Detecting and acting on the player's keyboard inputs.

2 Creating assets
Taking your sprites from the Library and into the game world proper.

3 Moving sprites
Making the bullets fly in the direction in which you fired them.

4 Detecting collisions
Reacting to two objects coming into contact.

5 Boundary checking
Defining what to do when an object is about to leave the play area.

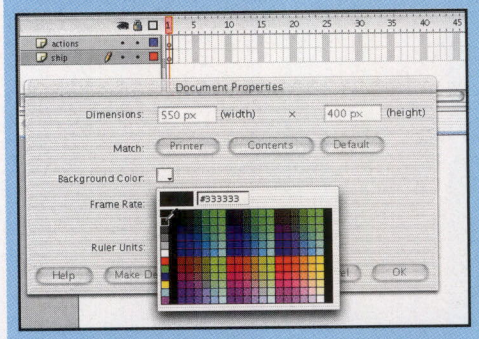
All of these structures are described in a modular fashion. This means that you can use them in the *Asteroids*-style game which we've provided code for in this feature, or you can take the structures out and plug them into any other games that you create.

Expertise provided by Kristian Besley and Ben Renow-Clarke.
[e] ken@pinderkaas.com or [w] www.pinderkaas.com

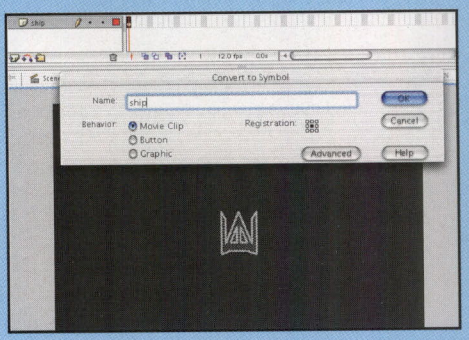
Files on disc
The files needed to support this tutorial can be found in the Tutorial\feat_action folder on the cover CD.

Illustration: Magictorch [w] www.magictorch.com

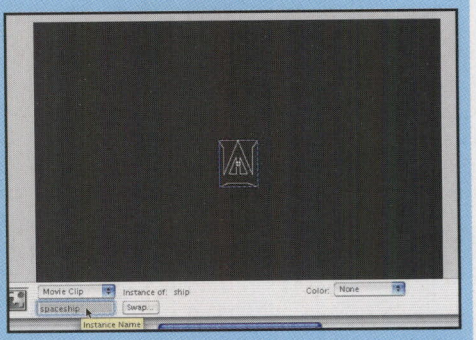
Preparing the asteroids movie



In a new movie, insert a new layer on the main timeline and rename the layers to Actions and Ship. Then choose Modify>Document and change the background colour to dark grey (#333333).

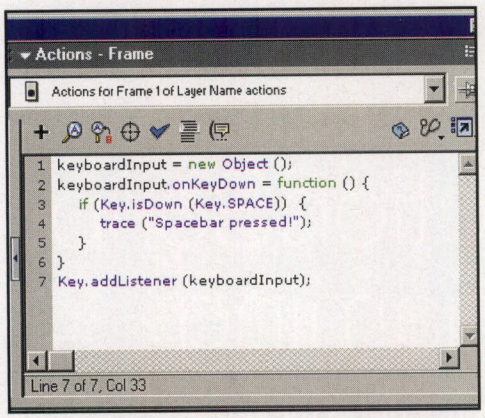


The next step is to draw a simple spaceship on the Ship layer and convert it into a movie clip symbol. Give your spaceship the name Ship and give it a central registration point.



Finally, simply select the Ship movie clip on the main stage and give it the instance name of Spaceship. Believe it or not, we've now prepared our entire movie for the asteroids!

1 Keyboard control: Listeners 01_keyListener.fla



Here is a typical script to send a message to the Output window when the spacebar is hit. Some example movement types are included on the CD.

Every game needs to detect some kind of input from the player, and the most common computer input device is the keyboard. *Flash MX* introduced keyboard Listeners for detecting input, which are far more efficient and processor-friendly than the old methods because they don't require a loop to check every frame.

In the *Asteroids* game, the motion is limited to simple left and right rotation only.

All of the code in this exercise will be on the same frame, so select frame 1 of the Actions layer and enter the following code:

```
MovieClip.prototype.keyboardControl = function(toMove, rotSpeed) {
kbdGet = new Object();
kbdGet.onKeyDown = function() {
    if (Key.isDown(Key.SPACE)) {
        trace ("Fire!");
    }
    if (Key.isDown(Key.RIGHT)) {
        toMove._rotation += rotSpeed;
    }
if (Key.isDown(Key.LEFT)) {
        toMove._rotation -= rotSpeed;
    }
}
Key.addListener(kbdGet);
};

//initialisation
keyboardControl (spaceship, 20);
```

This code defines a method called keyboardControl, which checks for input. When either of the cursor keys is pressed the ship is rotated, while a spacebar press displays "Fire!".

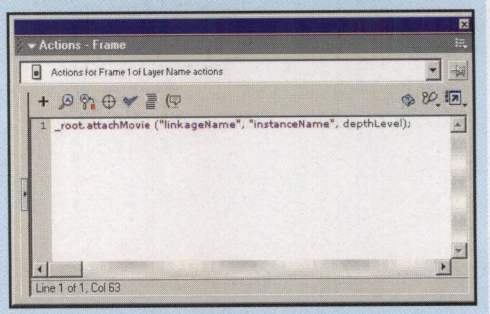
This method is then applied to the spaceship movie clip. In fact, this method can be applied to any movie clip by simply setting the instance name and rotation speed.

2 Attaching from the Library: attachMovie

02_attachMovie.fla

Whether it's another enemy or another weapon to defeat it with, creating new items is an essential part of all games.

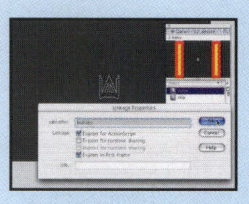
Before attaching a bullet, you need to draw one. Create a new movie clip symbol called Bullet and draw a bullet inside it. Centre it on the stage and then return to the main timeline.



In *Flash*, adding elements dynamically from the Library is done using attachMovie.

The Bullet movie clip now needs a Linkage name. Right-click on it in the Library and select Linkage. Tick the Export for ActionScript box and enter bullets as the Identifier, then click OK.

Back in frame 1 of the Actions layer, add the following code before everything else:



```
MovieClip.prototype.shootBullet = function() {
    if (_root.bulletOnScreen == false) {
        _root.attachMovie("bullets", "liveBullet", 1);
        currentBullet = _root["liveBullet"];
        currentBullet._x = this._x;
        currentBullet._y = this._y;
        currentBullet._rotation = this._rotation;
        _root.bulletOnScreen = true;
    }
}
```

This part of the method attaches a copy of the Bullet movie clip from the library, giving it the instance name liveBullet. The bullet is then positioned in the same place as the calling object.

In the keyboardControl method, change the code for the spacebar to:

```
if (Key.isDown(Key.SPACE)) {
        toMove.shootBullet ();
}
```

This will change the spacebar so it fires a bullet.

Add the following line of code to the very end of the script:

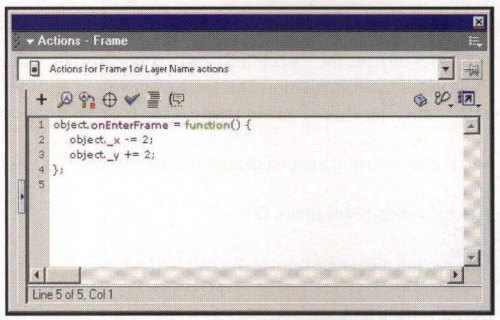
```
bulletOnScreen=false;
```

The bulletOnScreen variable is used to limit the number of bullets which appear on screen at any time to one.

Test the movie, and when you press the spacebar you'll see the bullet appear and mimic the ship's x, y and rotation.

3 Changing movie clip Properties: moving sprites 03_moveBullet.fla

Now that the bullet is on screen it needs to move a little, and a few additions to the shootBullet method will take care of this. Before we do this, look at the screen on the right. This shows the standard syntax to set the x and y properties of an object so that it moves consistently.



As a quick note here, object._y += 2 is just a shorthand way of representing object._y = object.y + 2, and the same goes for -= in the opposite direction.

Remove the last two braces from the shootBullet method and add the following code:

```
currentBullet.calculateSpeed (_root.bulletSpeed);
currentBullet.onEnterFrame = function() {
            currentBullet._y -= currentBullet.yadd;
            currentBullet._x += currentBullet.xadd;
            if ((currentBullet._x>_root.stageWidth) || (currentBullet._x<0) ||
(currentBullet._y>_root.stageHeight) || (currentBullet._y<0)) {
                currentBullet.removeMovieClip();
                _root.bulletOnScreen = false;
            }
        }
    }
}
```

This part of the method does two things — it moves the bullet and it also checks to see if it is still on the screen.

The calculateSpeed method called in the shootBullet method uses a little math to calculate the x and y speed according to its angle, filling the variables xadd and yadd with the results. These are then used in the shootBullet function to move the bullet according to its current position.

```
MovieClip.prototype.calculateSpeed = function(velocity) {
    this.xadd = velocity*Math.sin(this._rotation*(Math.PI/180));
    this.yadd = velocity*Math.cos(this._rotation*(Math.PI/180));
};
```

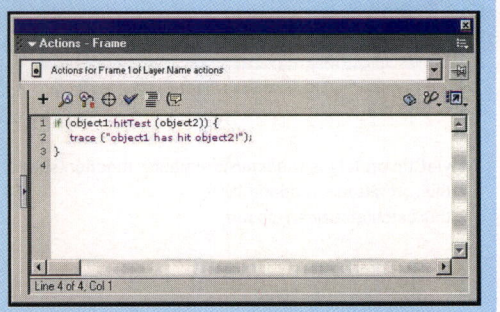
Add these variables to the very end of the code and then test the movie:

```
_root.bulletSpeed = 10;
_root.stageWidth = 550;
_root.stageHeight = 400;
```

This time the bullet speeds off and allows you to shoot again when it has left the edge of the screen.

4 Collision detection with hitTest 04_checkCollision.fla

The simplest method of detecting collisions in *Flash* involves using the bounding boxes that *Flash* places around all shapes. Unfortunately, what this means is that, in *Flash*'s eyes, all your circles become squares.

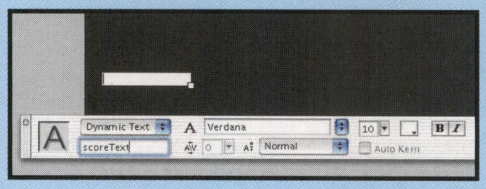


The standard Flash detection method is called hitTest. This common syntax checks if object1 has hit object2.

Insert two new layers, one called Text and one called Asteroid. On the Asteroid layer draw a large rock, making it more square than circular so that it will work satisfactorily with our collision detection method. Convert it to a movie clip of the same name with a central registration point and give it an instance name of Asteroid.

Now create a dynamic text box on the text layer and give it the instance name of >

scoreText. This text box will be used to keep score every time one of the ship's bullets hits an asteroid. The value of this textbox is set using the following code:




```
scoreText.text = 50;
```

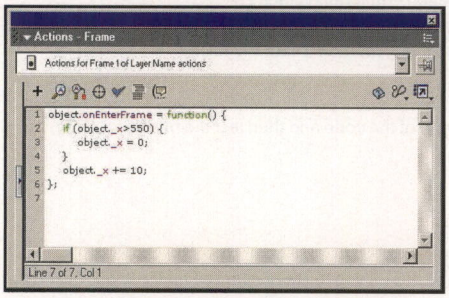
The next function is used to increment the score. Insert this script at the very start of the code:

```
keepScore = function (addScore) {
    _root.totalScore += addScore;
    _root.scoreText.text = "Score: "+_root.totalScore;
};
```

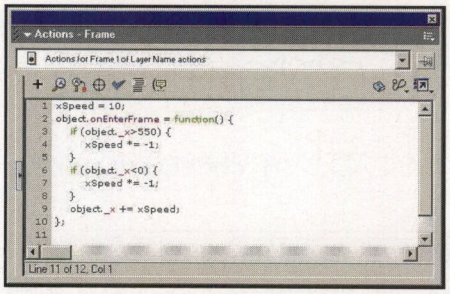
Now add the code below. The checkCollision method performs a hitTest with the parameters sent to it. If true, it tells it which function to perform. The enterFrame code for the asteroid simply runs hitFunction when there is a hit, making sure that the score is incremented.

```
MovieClip.prototype.checkCollision = function(toHit, functionToRun) {
    if (this.hitTest(toHit)) {
        functionToRun(this);
    }
}
hitFunction = function () {
    trace ("the bullet hit the asteroid!");
keepScore (100);
}
asteroid.onEnterFrame = function () {
    this.checkCollision(_root.liveBullet, hitFunction);
}
```

5 Boundary Checking: if statements 05_asteroidMovement.fla



Looping syntax: exit stage left, enter stage right, exit left...



Bouncing syntax: *= -1 here reverses the previous value (from 5 to -5 back to 5).

A static asteroid isn't much threat to the galaxy so you need to make it really hurtle through space. In this *Flash* file, however, the galaxy is only 550 x 400 pixels, so fast-moving asteroids would rapidly disappear. To counteract this, we can make the asteroids bounce off the sides of the screen or loop over to the other side. The most basic syntax for each method is found in the screens shown on the left.

We're using the looping syntax for our game, which means you need a method to move the asteroids and to provide the checking. The way in which we do this is by removing the entire asteroid.onEnterFrame function and adding the following code before everything else:

```
MovieClip.prototype.randomMovement = function(speed) {
    this._rotation = random(360);
    this.calculateSpeed (speed);

    this.onEnterFrame = function() {
        this._y -= this.yadd;
        this._x += this.xadd;
        if (this._y>stageHeight) {
            this._y = 0;
        }
        if (this._y<0) {
            this._y = stageHeight;
        }
        if (this._x>stageWidth) {
            this._x = 0;
        }
        if (this._x<0) {
            this._x = stageWidth;
        }
        this.checkCollision(_root.liveBullet, hitFunction);
    }
}
```

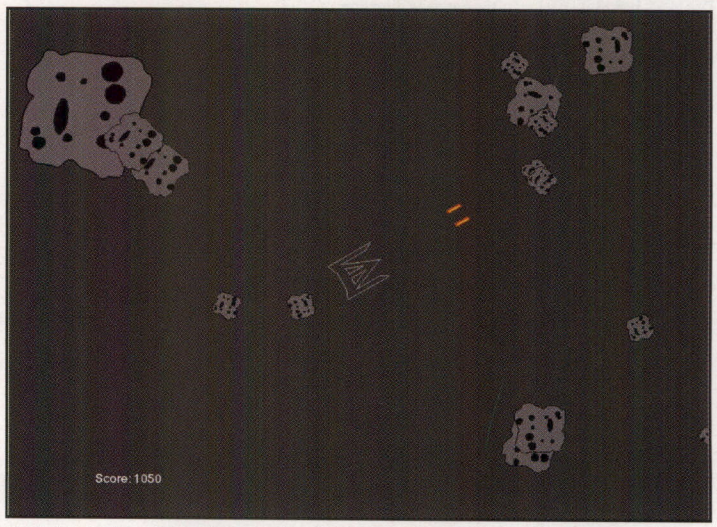
This function uses the calculateSpeed method to give the asteroid a direction based on an initial random rotation. The asteroid is moved and the boundaries are checked each frame. The checkCollision code is run to see if it has collided with a bullet.

Add the following at the end of all the code to move the asteroid at a speed of 2.

```
asteroid.randomMovement (2);
```

Now run the movie and aim at the asteroid!

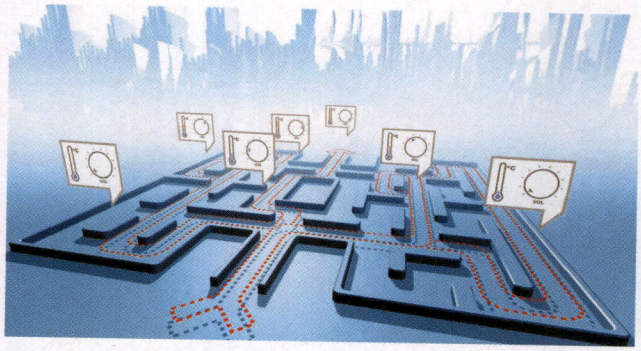
You can take a look at asteroids.fla on our cover CD to see the final code which is required to complete the game. CD-S



The finished thing, in all its retro arcade glory. The techniques shown in this tutorial are modular and so they can also be applied to a number of other games, so you can use them to experiment.

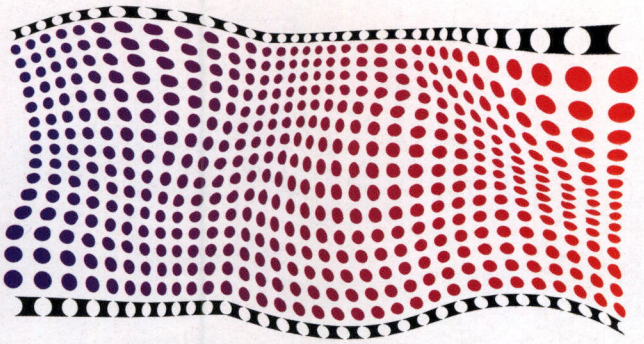
Technical skills

Create your online gaming masterpiece with our in-depth guides to getting the best out of all aspects of game design...



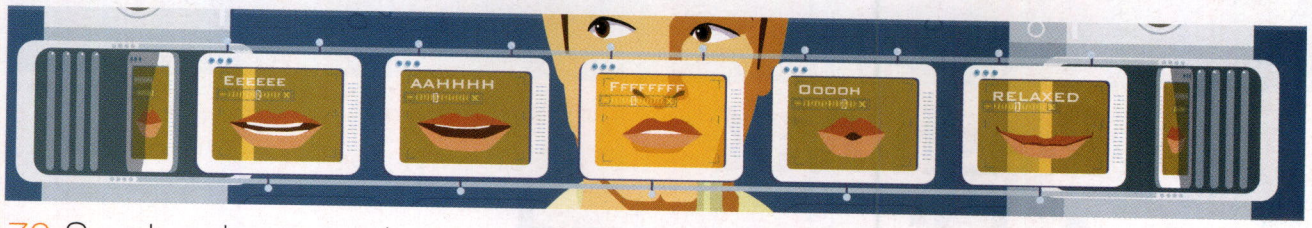
62 Add sound effects

Audio is an often overlooked part of the online gaming experience. Friends of Ed author Jerome Turner explains how to make it an integral part of your design and shows how you can create a unique, sound-based game



66 Use video clips in Flash

The ability to import and edit video clips in *Flash MX* means you can deliver video content to almost anyone online. Karl Hodge explains how you can create a whole game using interactive video footage

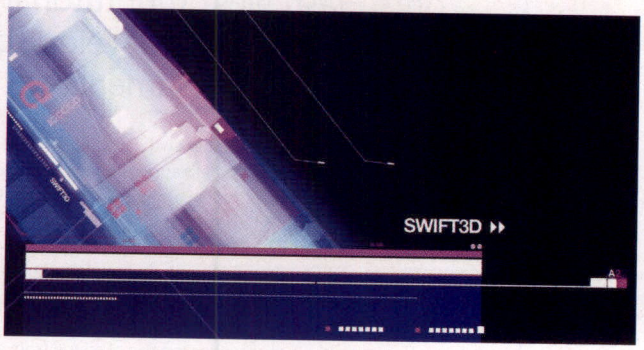


70 Synchronise speech

Our comprehensive guide to lip-synching using *Flash Amp* means you can get word-perfect game and cartoon character animation the fast and easy way. Jerome Turner's tutorial shows you how

74 Create a preloader

Once you've designed your game, chances are it'll be quite large. Chris Schmidt shows how you can add a user-friendly preloader so gamers know exactly how long they'll have to wait before they can start playing



78 Make a 3D robot

We opted for a fearsome, claw-waving robot, but whatever character you want to use in your games, George Cairns's introduction to *Swift 3D* explains the essential elements that you'll need to master

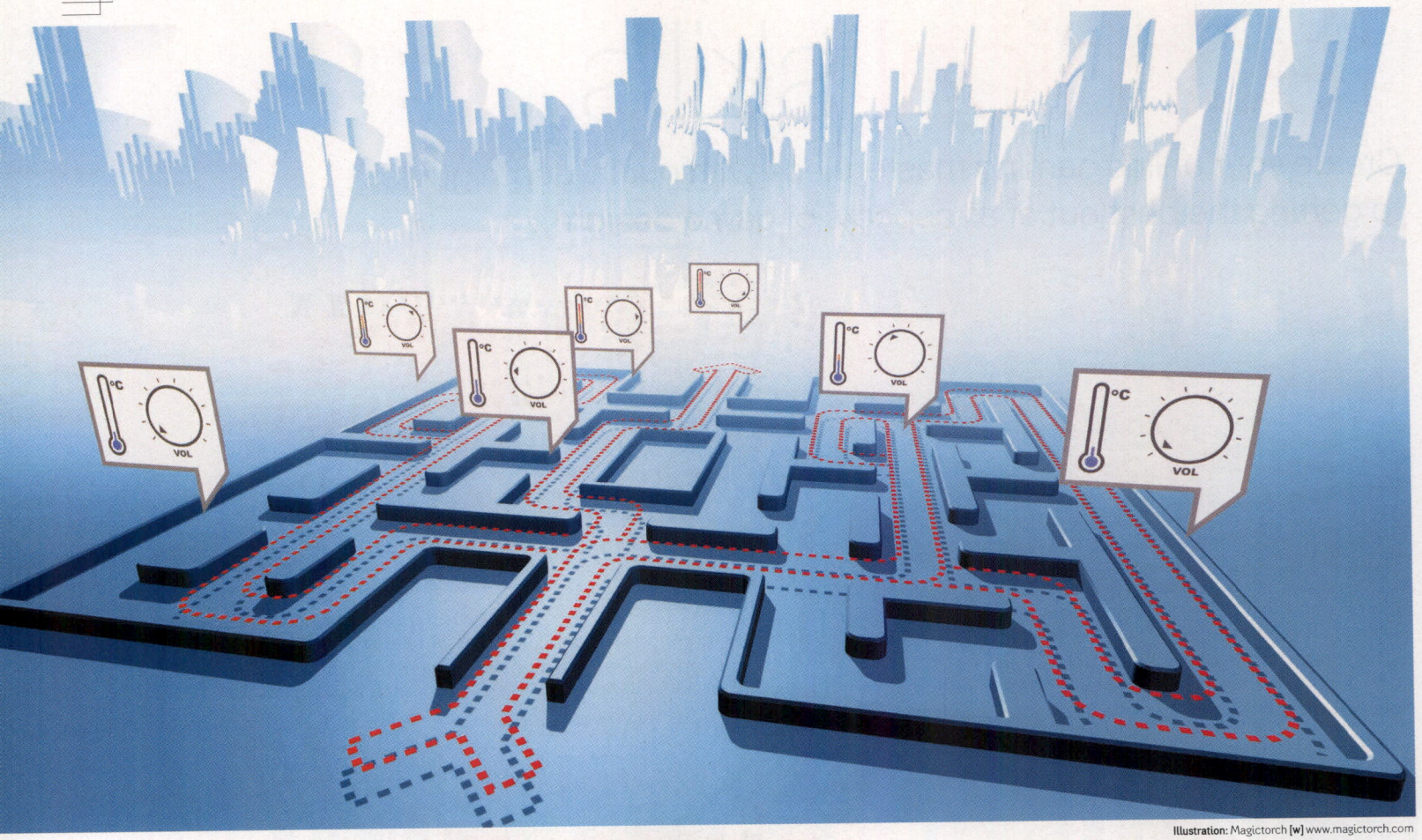


Illustration: Magictorch **[w]** www.magictorch.com

Flash MX
Adding sound

Bored of seeing audio clips clumsily tacked onto *Flash* files? Our tutorial unlocks the new *MX* capabilities to develop sound as an inherent part of gameplay…

Expertise provided by Jerome Turner. Jerome is a Friends Of Ed author and has worked on several books including *Foundation iMovie 2*, *Foundation Dreamweaver MX* and *Flash MX Studio*.
[w] www.jerometurner.co.uk,
[e] info@jerometurner.co.uk.

Files on disc
The files that accompany this tutorial can be found in the Tutorial\tech_sound folder on the cover CD.

Most *Flash* designers will have used sound in their work somewhere along the line, but what happens once you tire of simply dragging various 'pings' and 'bongs' onto your buttons? In the same way that sound recordists in film often lose out to the cameraman in terms of recognition for their work, the potential of audio on the Web is often neglected.

We start by showing you how sound can be recorded and prepared before importing to *Flash*. Then we take a look at how clips work as part of a *Flash* movie to be manipulated using ActionScript.

The game itself approaches the use of sound from a new angle. *Richard The Cat* started off as a generic maze hide-and-seek game (CD file soundtut_01.fla) but this tutorial shows how to develop the idea with ActionScript to incorporate sound as an intrinsic part of the game-playing strategy. When was the last time you played a *Flash* game that required the use of two of your senses?

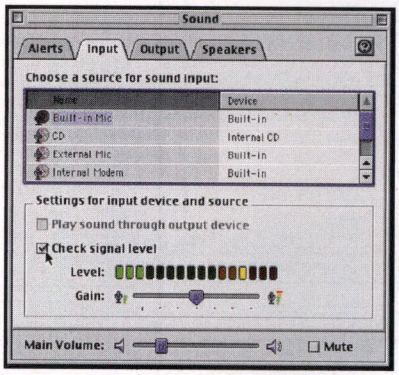
Maybe the next version of *Flash* will have smellovision as well, but for now we start with the game as a half-finished fla, ready to be beefed up with the addition of our audio files.

Part 1: Creating sound files

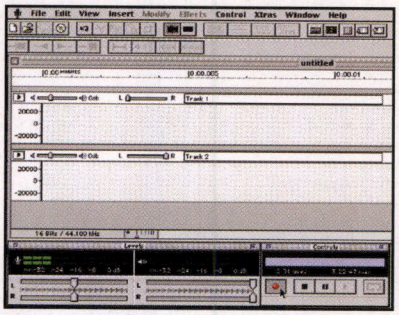
The first thing to do is record a demonic laugh…

No program?

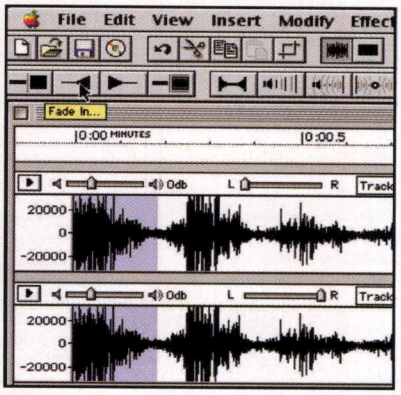
If you don't have a sound editing package like *Sound Edit* or *Peak* to create your own sound files, you'll find all the sounds used in this tutorial on our cover CD in the Tutorial\tech_sound folder.



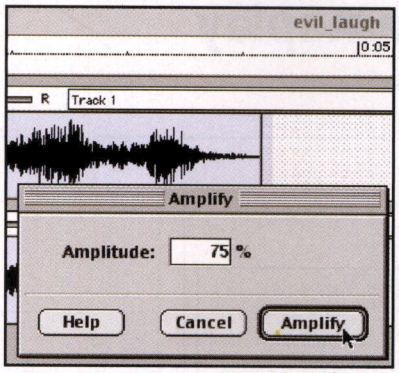
1 Make sure your computer's sound input is set to record using a plugged-in or built-in microphone, depending on what's available to you, then open your sound application.



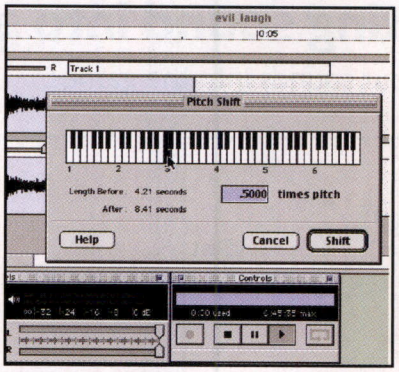
2 To create your own version of evillaugh.wav (which can be found on the cover CD), start a new file, click Record (Window>Controls) and laugh demonically into your microphone. Hit Stop when you've finished but remember not to get too carried away – the longer the sound, the larger the file is going to be in *Flash*.



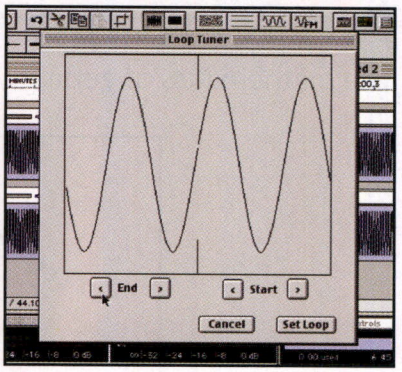
3 Select any 'silent' areas at either end of your recording and trim them. It the sound starts or ends too harshly, cover this area and Fade In or Out.



4 If the waveform has lots of contrast between loud and quiet areas, use a 'normalising' effect to centre the volume and then amplify the sound to around three quarters of the maximum level.



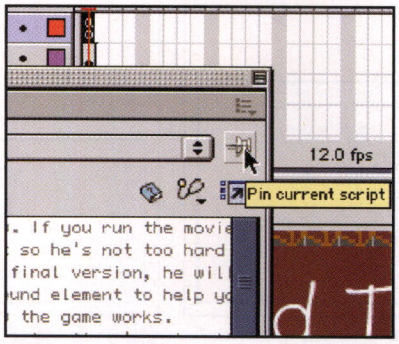
5 To add an extra element of creepiness to the sound, change the pitch or duration to slow down the laughter, then Save As either a WAV or AIFF file for use in *Flash*.



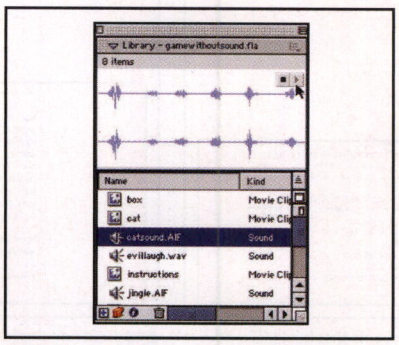
6 To create soundtrack.wav, open JS1010.aiff from the Sound Library files and trim a short looping sequence. Ensure the playback controls are looping, Copy and Paste into a new file and Save.

Part 2: Sound Objects in Flash MX

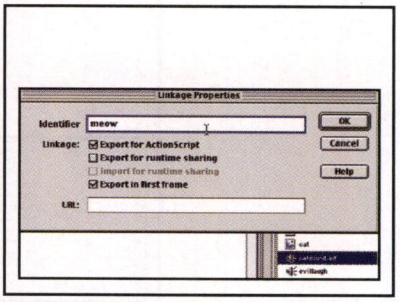
Make a meowing Sound Object…



1 Start by opening the soundtut_01.fla in *Flash MX* and read through the comments to guide you through the ActionScript on Frame 1. This is the location for all the code, so Pin the script to make sure the Actions window is always open on this frame.



2 Go to File>Import to Library and select catsound.aif, jingle.aif (also available from the Sound Library) and evillaugh.wav. You will see them listed in the Library panel where they can also be played back and renamed if necessary.



3 ActionScript can refer to sound files in the Library without placing them on the stage, but to do this we need to give each one a Linkage Identifier. Select the catsound.wav file and then Linkage from the Library's drop-down menu. Tick Export for ActionScript and name the Identifier. Repeat for each file.

Sound Objects in Flash MX continued...

Sound objects

Using Linkage Identifiers, rather than dragging clips onto the stage, means that your work area is more accessible and uncluttered. However, as Sound Objects can seem quite abstract (you can't see them) you might like to think of them as drawers with each one containing a sound file from the Library. Any change or reference to the Sound Object is like opening a drawer and amending the file inside.

```
9  //
10 //This displays the intro.
11 intro.gotoAndStop(1);
12 catDetect = new Sound();
13 catDetect.attachSound("meow");
14 win = new Sound();
15 win.attachSound("jingle");
16 lose = new Sound();
17 lose.attachSound("dead");
18 //This creates a new Object used to
19 //and plays all the rest of the act
```

4 After the first line of ActionScript (Line11), add the commands shown. This creates a Sound Object called catDetect and attaches the audio file via its Identifier "meow". Repeat to make Sound Objects 'win' and 'lose', which will wait here until they're 'called' later in the script.

```
20 start = new Object();
21 start.onKeyDown = function() {
22     if (Key.isDown(Key.SPACE)) {
23         intro.gotoAndStop(2);
24         restartText = "HIT SPACE TO RESTART";
25         statusText = " ";
26         //This initializes the cat and man positio
27         //Note the cat is placed ON TOP of the maz
28         man._visible = true;
29         cat._visible = true;
30         man._x = 10;
31         man._y = 410;
32         cat._x = 10;
33         cat._y = 390;
34         catDetect.start(0, 9999);
35         //This registers the time in milliseconds
36         startTime = getTimer();
37         //All the remaining code loops until the g
38         //been restarted by hitting SPACE.
39         _root.onEnterFrame = function() {
40             // MOVING THE MAN
```

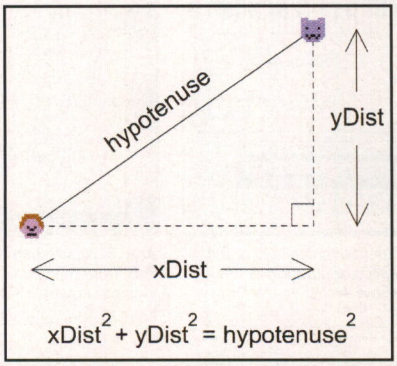
5 Pressing the space bar starts the game. The player then needs to hear the cat meowing, so apply a 'start' command to the catDetect object (line 34). The two parameters (0, 999) specify the point at which the sound should start playing, in seconds, and the number of times it should loop.

```
25         cat._y = 390;
26         //The cat sound is stopped
27         // This avoids the sound be
28         //due to it's repeated loop
29         catDetect.stop();
30         catDetect.start(0, 9999);
31         //This registers the time i
32         startTime = getTimer();
```

6 However, if the space bar has been pressed to restart the game for any reason, catDetect will also start again, on top of the last time it was called, so we check the sound isn't playing each time with a stop command (line 29) placed before the start.

Part 3: Incorporating sound into gameplay

The mathematics of getting your sounds into the game...



1 Our first instance of incorporating a sound into the gameplay comes in the form of the cat meowing – the closer the man is to the cat, the louder the sound. We need to use Pythagoras' theorem to work out the distance of the man to the cat.

```
68          //starting the loop here.
69          if (maze.hitTest(cat._x, cat._y, true)) {
70              cat._x = 10+(Math.round(Math.random()*19))*20;
71              cat._y = 10+(Math.round(Math.random()*19))*20;
72          }
73          // This calculates the distance from the man to the
74          // in terms of it being the hypotenuse of a right a
75          // triangle given the rule that
76          //xDist squared + yDist squared equals hyp squared.
77          xDist = cat._x-man._x;
78          yDist = cat._y-man._y;
79          hyp = Math.sqrt((xDist*xDist)+(yDist*yDist));
80          // This checks to see if the man has 'found' the cat
81          // and deletes the looping gameplay.
82          if ((man._x == cat._x) && (man._y == cat._y)) {
83              statusText = "YOU FOUND RICHARD!";
84              restartText = "HIT SPACE TO PLAY AGAIN";
```

2 In ActionScript terms the same rule looks like the above. In line 77 xDist is worked out by taking the man's x position from the cat's, and the same applies to yDist. The hypotenuse is found in line 79 using Math.sqrt to find the square root of our equation.

```
77          xDist = cat._x-man._x;
78          yDist = cat._y-man._y;
79          hyp = Math.sqrt((xDist*xDist)+(yDist*yDist));
80          // This checks to see if the man has 'found' t
81          // and deletes the looping gameplay.
82          if (hyp==0) {
83              statusText = "YOU FOUND RICHARD!";
84              restartText = "HIT SPACE TO PLAY AGAIN";
85              cat._visible = true;
86              man._visible = false;
87              delete this.onEnterFrame;
88          }
89          //This compares the start value of the timer t
```

3 In our original code for the game we had to use the man's x position and y position against the cat's values to check for a 'hit' (see line 82 in previous screenshot) but now we can check the hyp value instead (line 82 here) so change it accordingly.

```
79          hyp = Math.sqrt((xDist*xDist)+(yDist*yDist));
80          // This checks to see if the man has 'found' the cat
81          // and deletes the looping gameplay.
82          if (hyp == 0) {
83              statusText = "YOU FOUND RICHARD!";
84              restartText = "HIT SPACE TO PLAY AGAIN";
85              cat._visible = true;
86              man._visible = false;
87              catDetect.stop();
88              win.setVolume(100);
89              win.start(0, 1);
90              delete this.onEnterFrame;
91          }
92          //This compares the start value of the timer to the c
```

4 In addition to the events that take place on finding the cat, add the lines 87-89. As the cat has been found, he can stop calling out for help. The volume for our win sound needs a parameter from 0-100, then start the sound as usual, looping it just once.

```
80          // This checks to see if the man has 'found' the
81          // and deletes the looping gameplay.
82          if (hyp == 0) {
83              statusText = "YOU FOUND RICHARD!";
84              restartText = "HIT SPACE TO PLAY AGAIN";
85              cat._visible = true;
86              man._visible = false;
87              catDetect.stop();
88              win.setVolume(100);
89              win.start(0, 1);
90              delete this.onEnterFrame;
91          }
92          if (hyp<300) {
93              meowVol = 100-(hyp/3);
94          } else {
95              meowVol = 0;
96          }
97          //This compares the start value of the timer to
```

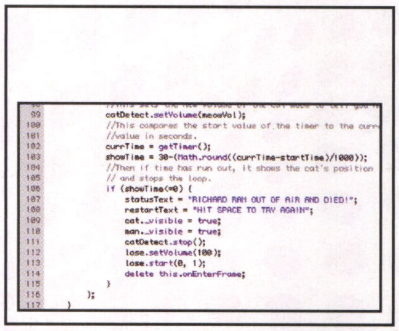
5 Next add the following 'if' statement which translates hyp into meowVol, a value between 0 and 100 to set catDetects volume. If hyp was larger than 300, the equation in line 93 would return a negative value so line 95 makes sure it can't drop below 0.

```
91          //The 'if' statement means that no
92          //set this, as meowVol would then
93          if (hyp<300) {
94              meowVol = 100-(hyp/3);
95          } else {
96              meowVol = 0;
97          }
98          //This sets the new volume of the
99          catDetect.setVolume(meowVol);
100         //This compares the start value of
```

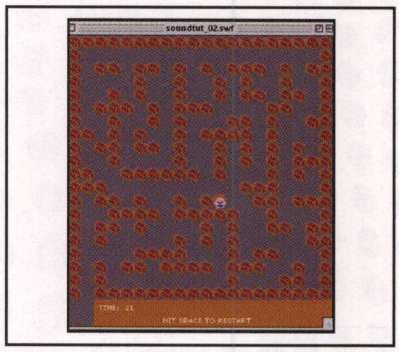
6 Finally, after all the maths, we have a value that can be used to setVolume of catDetect. As this section of the code loops from the moment the space bar is pressed, the cat meow's volume repeatedly updates to always tell you how close you are to the cat.

Compress in MX

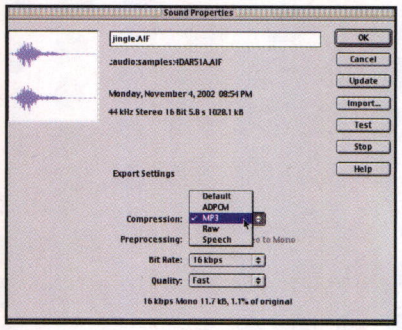
As most *Flash* games are aimed at the Web, MX allows us to compress the audio files used in order to keep file size down. Select your sound in the Library and choose new Compressions via the Properties panel. Think carefully about where you can sacrifice sound quality though, and try exporting with different settings to compare quality over file size.



7 The game is already set to end if the player runs out of time, but as they've spent all that time looking, add line 109 to show them where he was hidden. Also stop the catDetect sound as before. Finally setVolume and start the lose sound object to play it once.



8 Now test your fla and marvel at how much more the use of those sounds has brought to your game. If there are any problems, check your finished file's code against soundtut_02.fla, using the comments as a guide.



9 If the exported .swf is too large for your intended Web audience, don't forget you can compress the audio files used in *Flash*. In the Library select a sound, choose Properties from the drop-down, change the Compression type and click Update. Test again to note the difference in file size.

Part 4: Controlling sound

Create a soundtrack that plays throughout the game…

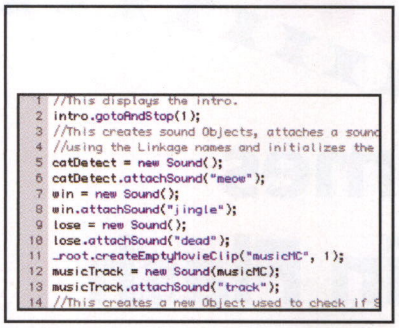
Using independent sounds

If you create a Sound Object without specifying a movie clip for its parameter, any methods used on it will affect the global sounds of the fla. Although this won't be problematic where one sound follows another, if more than one sound is playing at the same time with different volume or pan settings, parameters will need to be specified in the Sound Object. For example:

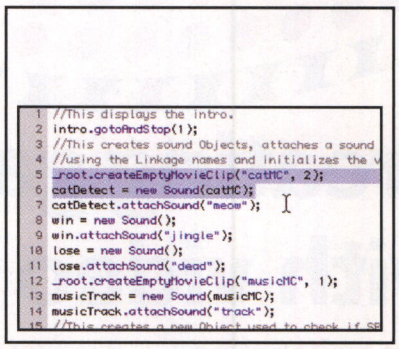
```
_root.createEmptyMovie
Clip("mc",1);
mySound = new Sound(mc);
mysound.setPan(20);
```

No preload

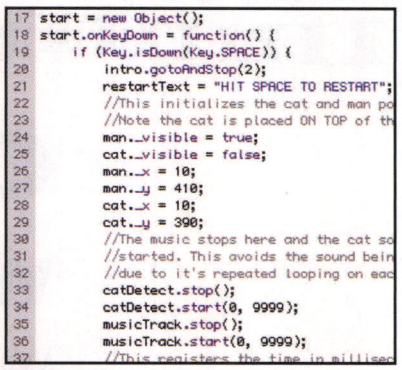
If you're thinking this game could do with a preloader, then think again. To start with, the file itself isn't that large, but more to the point any file attached using an Identifier (rather than appearing on the stage) loads up before Frame 1 of your movie. So if you did include a preloader on Frame 1, all the sound would have already loaded up anyway so it becomes useless.



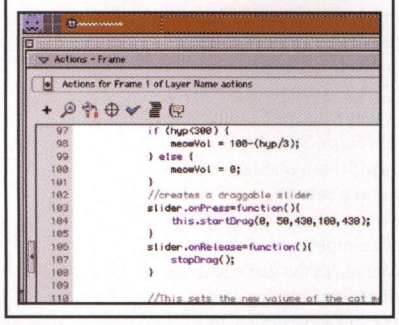
1 Although it's not particularly helpful in the cat game (where we need to listen for the meows), we'll finish by creating a soundtrack that plays during gameplay. Import the soundtrack.aif with the Identifier track. This sound will be controlled independently of the global sound, so specify it to a new Movie Clip (lines 11-13).



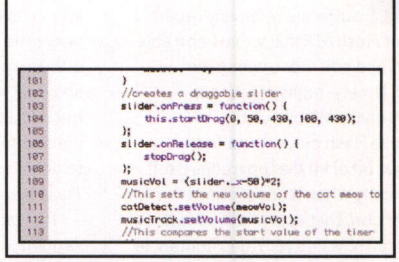
2 As catDetect will be playing at the same time, it also needs to be specified to its own new Movie Clip to distinguish it. Otherwise any command sent to catDetect (such as setVolume) will affect all sounds. Note the second parameter, positioning this empty movie clip on level 2.



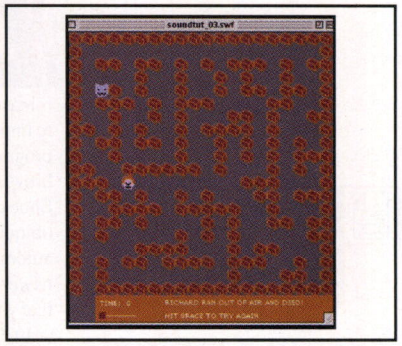
3 As before with catDetect, place a stop then start after if (Key.isDown(Key.SPACE)) statement to initialise the music at the start of the game. Again, this ensures that the music doesn't overlap itself each time the game restarts.



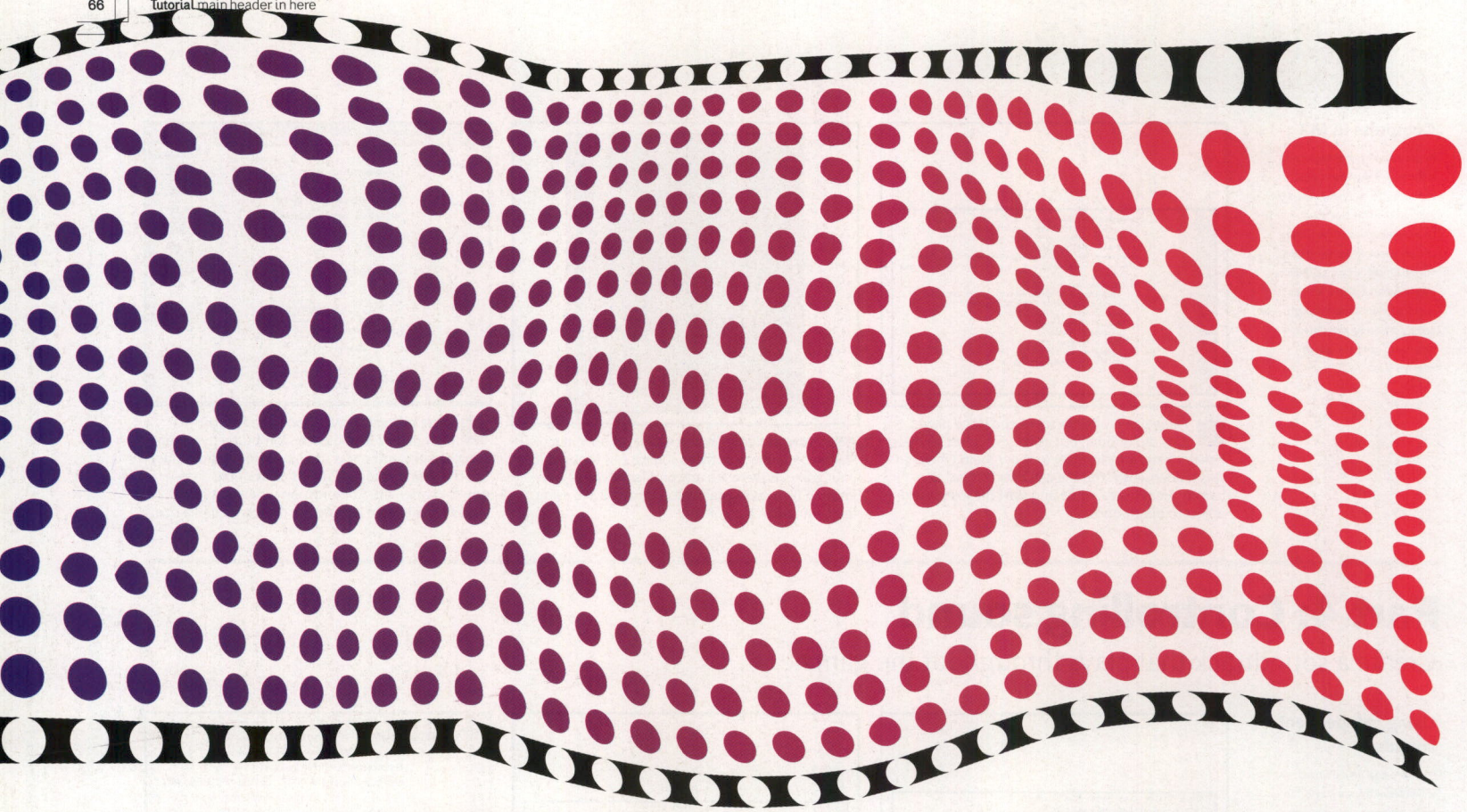
4 Next draw a small square movie clip on the stage, with the instance name slider. Place it on the left end of a 50 pixel horizontal line and add the following script to create a draggable slider.



5 Then create a variable from the slider's x position, which returns a number between 0 and 100 for musicVol (in line 109). Use setVolume to update the volume of the music as set by the player. Finally stop() the music in the if(hyp==0) and if (showTime<=0) statements, as we did earlier with catDetect.



6 The final file can be found on the CD as soundtut_03.fla, but consider other ideas that could be applied to your own games. Remember we've only touched on a few uses of Sound Objects – there's a lot more to work with in the Objects> Sound section of the Reference panel. ca>s



Creating games with video in Flash

Want to deliver video to any browser? With some help from *Flash MX* and an ordinary deck of cards, we'll show you how...

Although Macromedia *Flash* embraced most other media from the word 'go', it's only since the relatively recent release of *Flash MX* that we've been able to import and edit video within the program. Ninety-eight per cent of browsers on the Web have the Shockwave Flash player installed, a ubiquitous bit of kit that upgrades itself automatically when it encounters a newer version. That essentially means that you can now deliver video content to virtually anyone, regardless of their machine or their plug-in set-up. Much the same is true for the support offered to MP3 sound files in *Flash MX*.

In this tutorial, we've taken some video clips and, using some fairly basic *Flash* techniques, have woven them into an intriguing game. We've provided all the bits and pieces, so you won't need any extra hardware to take part. There's very little in the way of ActionScripting either; we wanted to concentrate on showing you how to get practical results quickly and easily from the Sorenson video import in *Flash*. Still, there's plenty to get to grips with here whether you're a *Flash* beginner or a veteran.

The game we've created uses video sequences of a simple card trick – the Three Card Monte. It's the game where a dealer shuffles three playing cards right in front of you and asks you to pick out a particular card. With a little Flash magic, we'll show you how the dealer always wins.

Flash tricks provided by Karl Hodge, who is a freelance new media journalist. You can contact Karl at **[e]** khodge@spodgod.com.

Files on disc

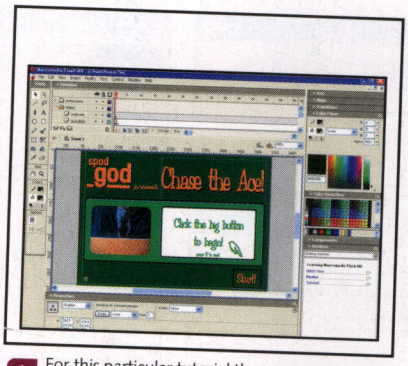
The files needed to support this tutorial can be found in the Tutorial\tech_video folder on the cover CD.

Part 1: Prepare your video clip

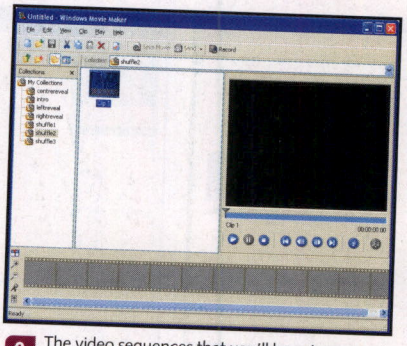
Before video clips can be loaded into *Flash*, they need to be prepared...

Favoured font

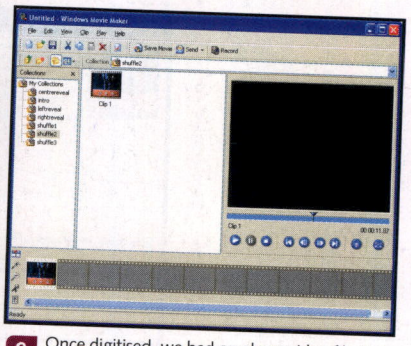
The public domain font Fairchild is used extensively throughout the movie. Before you begin, you need to install the TrueType font file provided in the Tutorial\Video folder on the cover CD. Both Mac and PC versions have been provided.



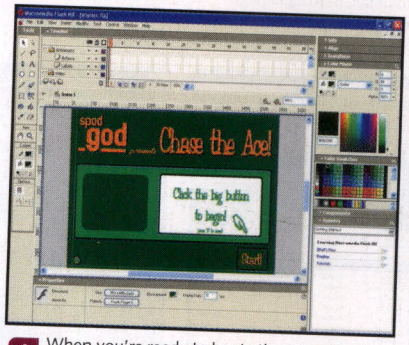
1 For this particular tutorial there was quite a lot of behind-the-scenes preparation. You'll be starting with most of the elements intact, but we'll briefly take you through how we did what we did, and tell you why we did it.



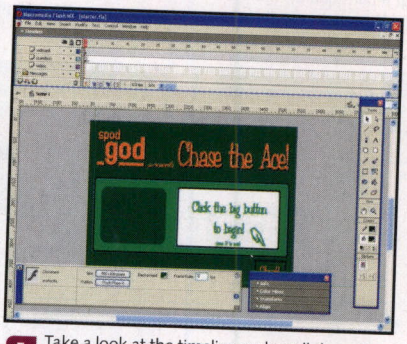
2 The video sequences that you'll be using were filmed using an analogue camcorder in one single take. We used the Dazzle Digital Video Creator, an analogue-to-digital video capture unit to bring the source material into a Windows PC. You could use a similar method, or use a digital video source.



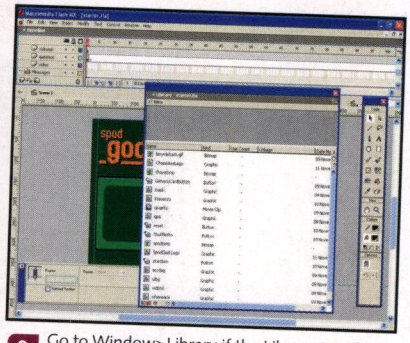
3 Once digitised, we had one long video file in MPEG format. It was imported into Windows Movie Maker, a simple video-editing package that ships with Windows XP. We used that to cut the video sequence down into smaller chunks, saving the results as a series of high-resolution AVI files.



4 When you're ready to begin the tutorial, start *Flash MX* and open the file 'starter.fla' from the cover CD. You'll notice that we've already done quite a bit of the work for you. The basic layout is already in place, as are a series of ready-named folders and layers.



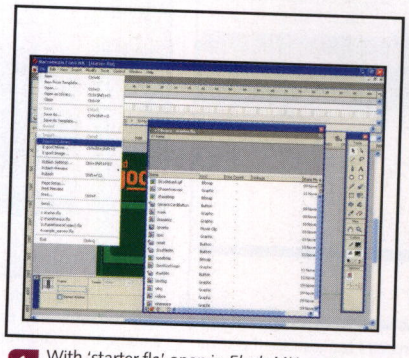
5 Take a look at the timeline and scroll down to see the folders and layers we've prepared. Several layers are already fully populated: layout, UIElements, messages, buttons, vidmask, question, reset. These contain animated elements that are present throughout the movie.



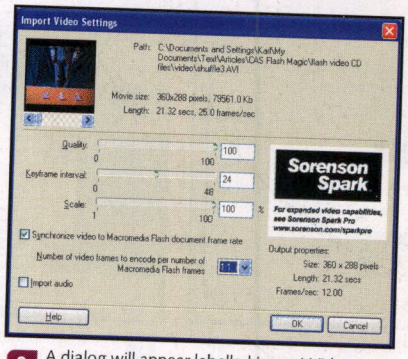
6 Go to Window>Library if the Library panel is not already visible. Expand the window so that you can see the Library items properly. Every single item in this movie is (or, in the case of the video clips, will be) a symbol. They're all ready to drag and drop into place.

Part 2: Add it to your Flash movie

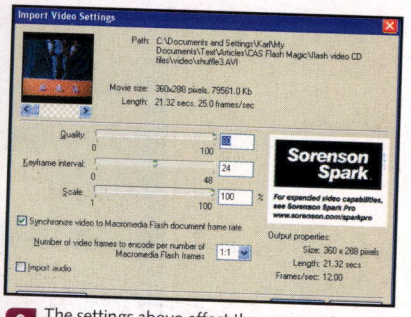
Here are the nuts and bolts of the project: importing, optimising and placing video clips...



1 With 'starter.fla' open in *Flash MX*, go to File>Import to Library. Browse to the folder called Video on the cover CD. You'll see seven Windows AVI format video files. Select them all and click on Open to begin importing them.



2 A dialog will appear labelled Import Video Settings. A preview of the movie appears in the top left corner – moving the scroll bar beneath plays it frame by frame. To the right of that there's information on the size and length of the video clip.



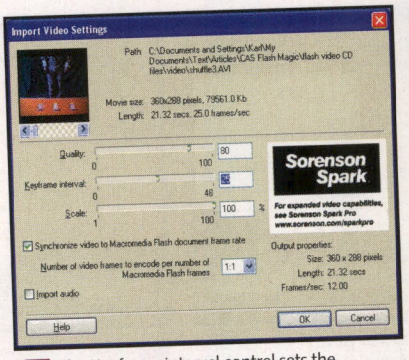
3 The settings above affect the compression and quality of your imported video. The Quality slider applies direct compression to the video clips. The lower the figure, the more compressed the video, but the quality begins to suffer noticeably in images with lots of transitions. We found that 80 per cent was an acceptable value for this project.

Transitions

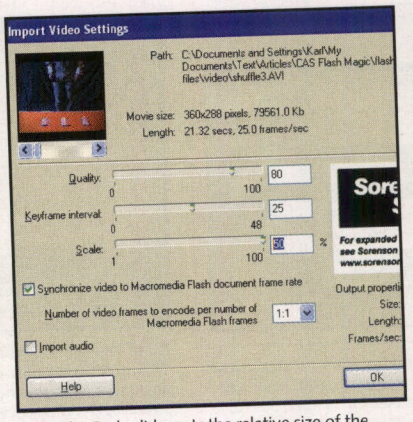
In *Flash MX*, video clips behave more like single objects than a series of frames. Still, you can rotate, skew and scale clips. You can also use tweening to animate the position of a video clip over the timeline.

Broadband capabilities

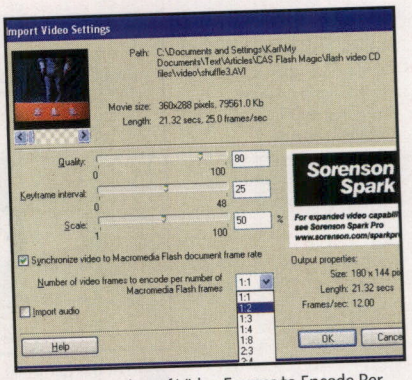
With Broadband connections becoming more usual, the optimisation options detailed here may not seem as important. However, you also have a hardware threshold to deal with – other people's machines may not be able to throw out the necessary processing power to deal with high-resolution video, animation, sound and interactivity as it streams down an ADSL connection. Keep this in mind as you author.



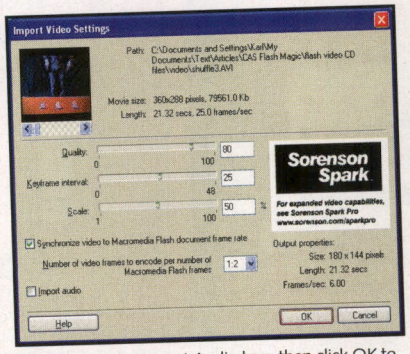
4 The Keyframe interval control sets the frequency of fully rendered frames per clip. For example, if you set the Keyframe interval to 25, every 25th frame is rendered as a full frame. The frames in-between simply record the differences between the keyframe. This is an effective compression method that retains quality. We used an interval of 25 – you could probably set it higher.



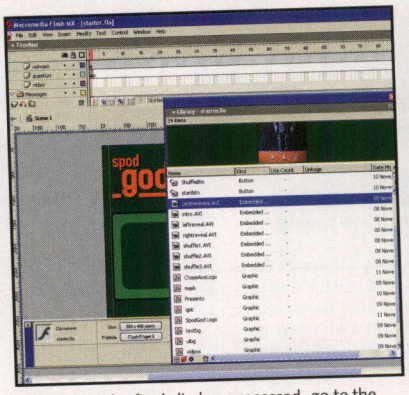
5 The Scale slider sets the relative size of the movie on import. The smaller the physical size of the movie, the smaller the eventual file size. Also note that the compressed detail in smaller images appears finer. We set this to 50 per cent.



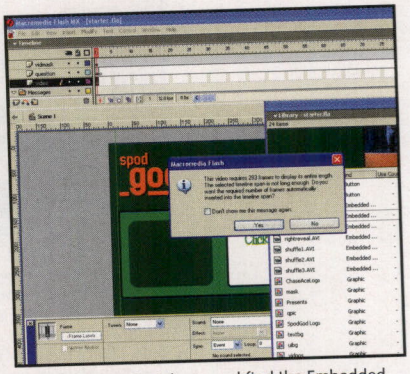
6 The 'Number of Video Frames to Encode Per Number of Flash Frames' setting lets you set up a ratio that enables further compression. We set it a 1:2, which means that every second frame of our movie was removed out and the preceding one doubled up. Setting the ratio too high will result in 'choppy' playback.



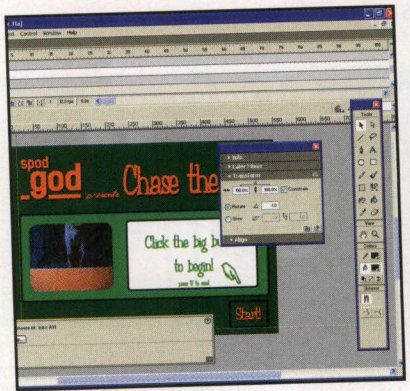
7 Untick the Import Audio box, then click OK to import the first video clip. *Flash* may take a couple of moments processing the clip before returning you to the Import Video Settings dialog so that you can tweak the settings for the next clip. *Flash* remembers all your setting except the Scale slider – so remember to put that to 50 per cent for all the clips.



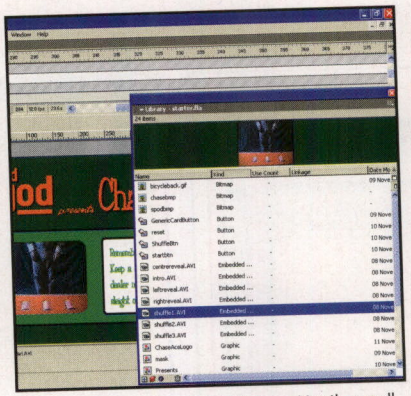
8 When the final clip has processed, go to the Library and click on the Kind column. The imported video clips should appear as Embedded Video. Go to the Timeline and find the video layer inside the Video folder. Click in the first frame and go to Insert>Blank Keyframe.



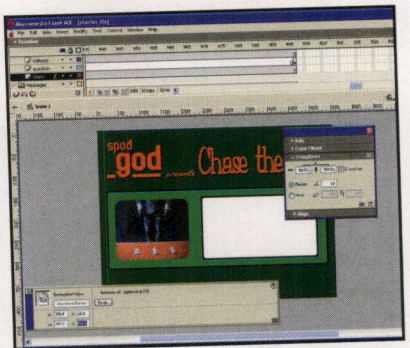
9 Return to the Library and find the Embedded Video clip named intro.AVI. With the first keyframe of the video layer still selected, drag and drop an instance of the clip to the stage. You'll be informed by *Flash* that the clip extends beyond the current number of frames and asked if you want to proceed. Click Yes.



10 Select the clip on the stage. You'll notice that the table in the image is slightly at an angle. Correct this by going to the Transform panel and setting the rotation of the clip to -1.0 degrees. With the clip still selected, go to the Properties panel and set its X position to 28.8 and its Y position to 158.4.



11 The first clip inserts 283 frames. Use the scroll bar at the bottom of the timeline to find the end of the clip and insert a new, blank keyframe at frame 284 in the video layer. Drag and drop an instance of the video clip shuffle1.AVI to the stage. Position it and correct the angle as before.



12 Continue to add the remaining video clips in the same way, each time allowing *Flash* to add a number of extra frames to the timeline, each time scrolling to the last frame of the clip in the timeline and adding a new, blank keyframe immediately after. Add the remaining clips in this order: shuffle2.AVI, shuffle3.AVI, leftreveal.AVI, centrereveal.AVI and rightreveal.AVI.

Part 3: Make it interactive

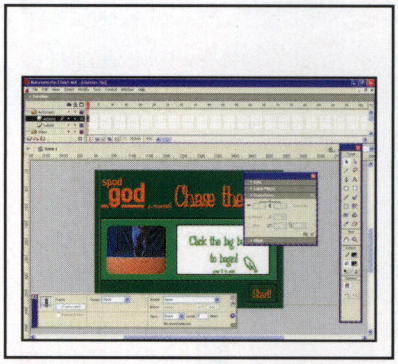
Add control to your videos, the *Flash MX* way…

Playing symbols

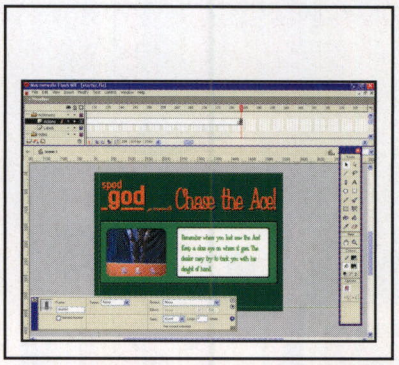
All the items in the movie are symbols, which means they can all be easily edited and updated without affecting the rest of the movie. It also means that you could easily change the overall look of this project without affecting the interactive elements.

It's magic!

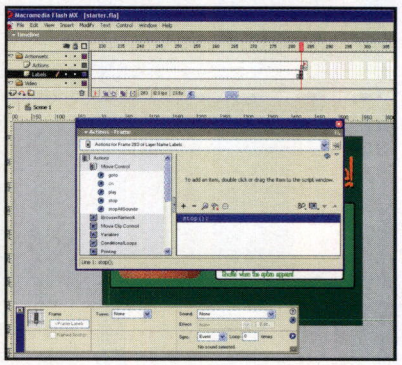
No – it's not really magic. Whichever card you select after viewing the shuffle, it will always be the wrong one. Based on a classic card illusion, our game substitutes sleight of hand for sleight of script...



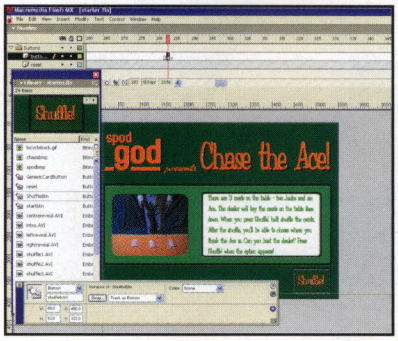
1 Once imported, Flash video can be manipulated like any other object. We prepare to add controls to our movie by laying down labels and adding simple 'stop()' actions at key points. Begin by going to the Actionsetc folder and adding a blank keyframe at the first frame of the Labels layer.



2 With the new keyframe selected, go to the Properties panel type in the label 'instructions' in the Frame text box. Add a keyframe at frame two and add the label 'intro'. Continue in the same way adding the label 'shuffle1' at frame 284, 'shuffle2' at 516, 'shuffle3' at 786, 'leftreveal' at 1042, 'centrereveal' at 1186 and 'rightreveal' at 1350.



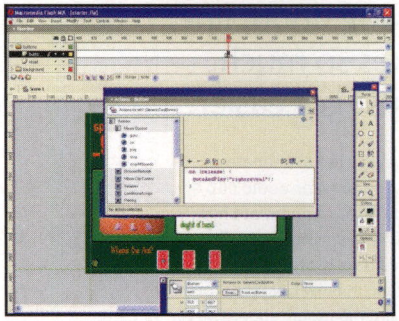
3 Go to the Actions layer and insert a blank keyframe in the first frame. Select the frame and go to Window>Actions. Add a stop() action to the frame – it's in the Actions>Movie Control section. In the same way, add blank keyframes with stop() actions to frames 283, 515, 785, 1041, 1185 and 1349.



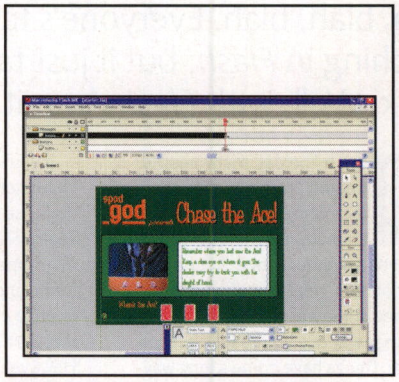
4 Those actions and labels will help us to control the video clips, with some help from a handful of buttons. Go to frame 283 in the buttons layer and insert a blank keyframe. Drag and drop an instance of ShuffleBtn from the Library to the stage, positioning it at X 450, Y 333. With the button selected, give it the instance name shufflebtn1 in the Properties panel.

5 Go to Window>Actions (if the Actions panel is closed) and, with the Actions panel set to Normal mode, double-click on 'goto'. Select the 'gotoAndPlay' line, then choose Frame Label from the Type drop-down. In the Frame box type shuffle1.

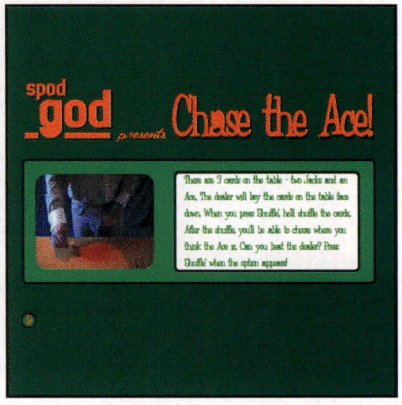
6 We've already added similar buttons for you at key moments throughout the presentation. The Shuffle button appears when the user has the choice of viewing a clip of video where the dealer mixes up the cards.



7 With the Actions panel still open, go to frame 515 in the button layer and select it. Notice that three playing card buttons appear on the interface, with a label saying Where's the Ace. Click on the stage to deselect everything, then select a button. Look at the Actions panel and notice that selecting one of the card buttons takes the user to the label 'leftreveal', 'centrereveal' or 'rightreveal' in the timeline.



8 If you look at the messages layer, you'll see that we placed text messages throughout the sequence at specific times to either comment on or explain the action that takes place in the video sequences. This technique could easily be adapted for use in video titling.



9 When you've finished, use Control>Test Movie to see how the finished game plays. Alternatively, you can open our finished version (chasetheace.fla) from the tutorial folder on the CD and try that. CD

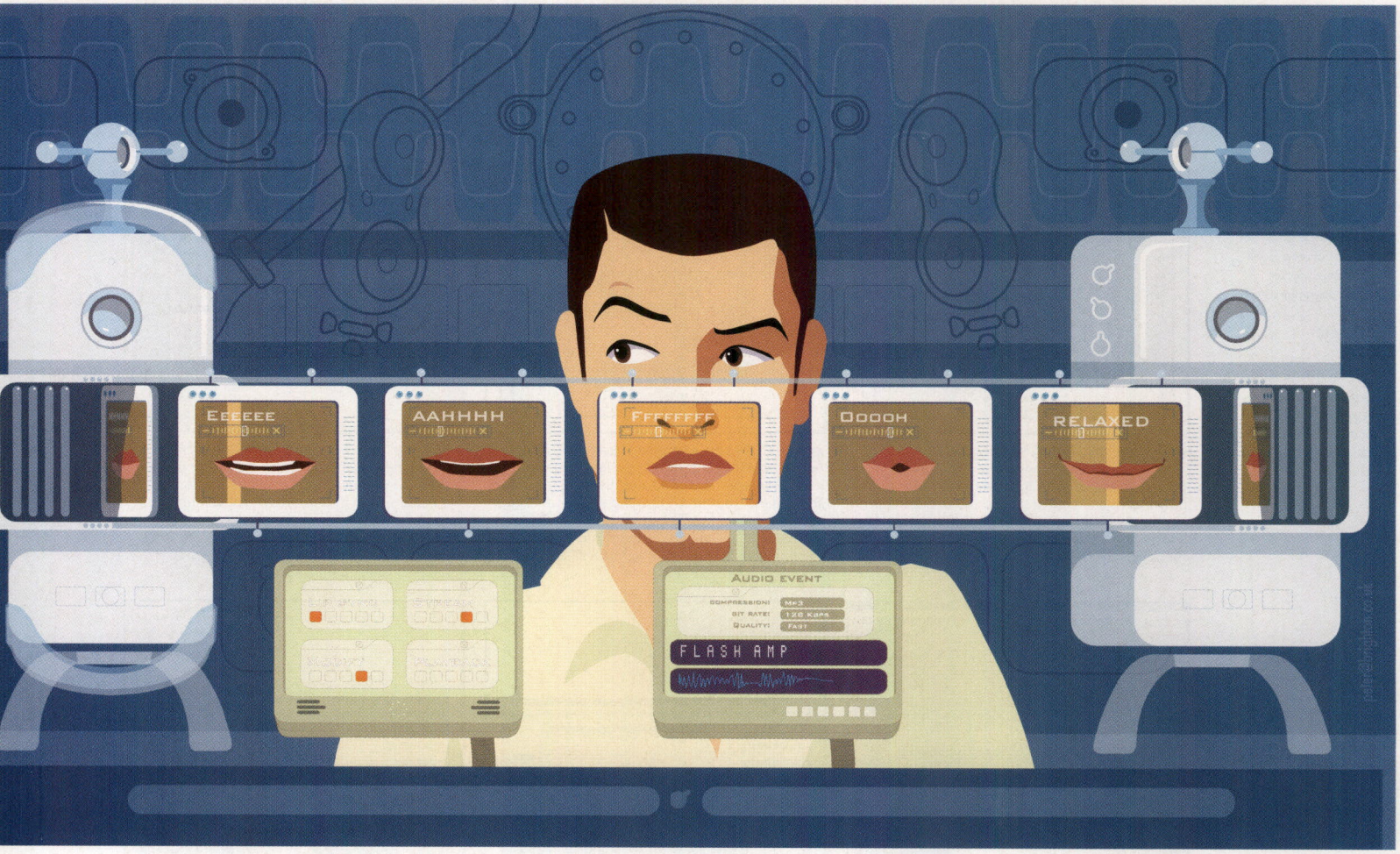


Illustration: Peter Greenwood **[e]** peter@brighton.co.uk

Lip-synching with Flash Amp

Blah, blah, blah. Everyone's talking about lip-synching in *Flash*, but it just takes so much work. Well, with *Flash Amp* it just got easier...

Expertise by Jerome Turner
[e] info@jerometurner.co.uk
[w] www.jerometurner.co.uk
Jerome is a freelance designer/writer and has worked on several Friends Of Ed books including *Foundation iMovie 2*, *Foundation Dreamweaver MX* and *Flash MX Studio*.

Files on disc
The files needed to support this tutorial can be found in the Tutorial\tech_cartoon folder on the cover CD.

Back in the day, many people treated *Flash* as a means of recreating traditional animation techniques such as stop-motion action. Draw a picture, draw another picture, repeat until madness sets in and someone discovers you scribbling away at a timeline as long as your arm. Things have come a long way since then. People are still creating the same kind of work but labour-saving devices such as *Flash Amp* combined with modern *Flash MX* ActionScripting can get the same job done in minutes.

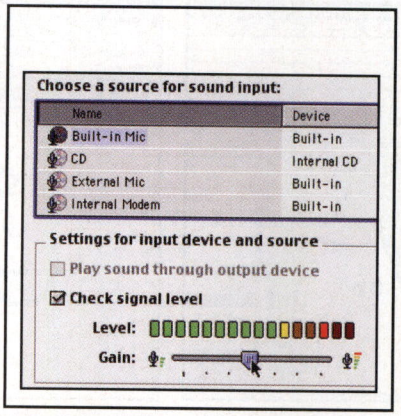
If you've ever tried to animate a character's lips along an audio clip of dialogue you'll soon appreciate how useful *Flash Amp* can be. Typically you might spend hours listening to each frame and then making decisions on how open or closed the mouth should be for that particular drawing in the timeline. However, using *Flash Amp* requires only ten drawings (keeping the file size down) and it automatically works out which one should be played according to the volume of the sound at each frame. Sounds clever? It is...

Part 1: Recording dialogue

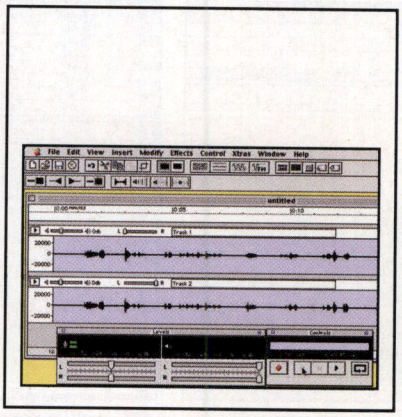
How to get the clearest conversation for use in your animations and cartoons...

Cheating Flash Amp

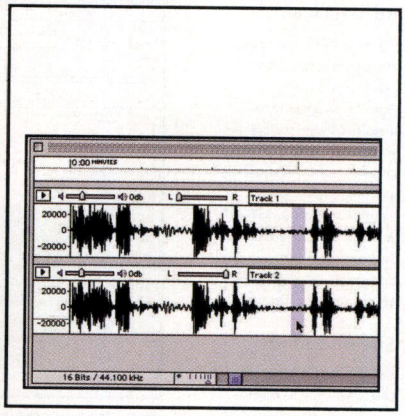
If there is too much noise in your sound file, you may find supposedly 'silent' areas hissing enough to return a 1 instead of a 0 in the amplitude list. If this is the case, there are two fixes. Try to clean up the audio in your sound app so it's quieter before processing again, or simply alter a few of the values in the ActionScript arrays.



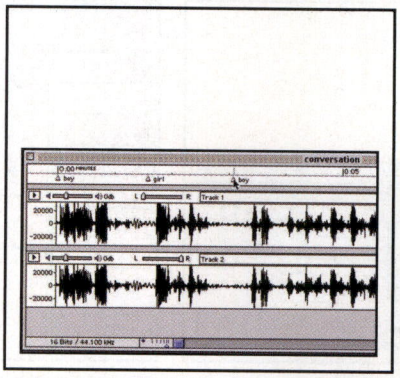
1 You'll find the voices.aiff file on the CD but feel free to record your own version. Write the script for a short conversation (four lines or less) between two people. Then make sure your audio input and recording levels are set and open your sound application.



2 Record the conversation as one take with slight pauses between each new line of dialogue. Make sure the readers sound 'animated' (in other words, not monotonous) to ensure a good range of volume as this will reflect in the *Flash Amp* processing later on.



3 Trim silent areas from either end of the sound and adjust any gain or volume controls to make sure the audio is peaking around the top of the scale. If it's too high, normalise and smooth to clear up any noise. If necessary, trim out any over-long silent areas between lines.



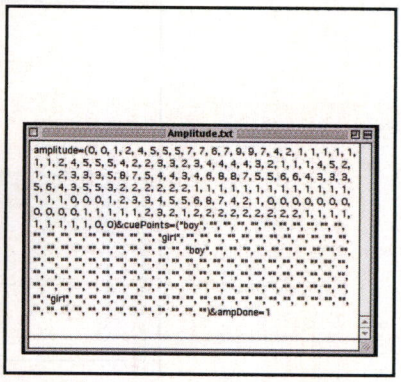
4 Insert Cue Points between 'lines' to distinguish the speaker – one before the boy starts talking, then another before the girl, back to the boy, and so on. Name them according to their characters as these will act as a switch telling ActionScript which one to animate. Save the file as an AIFF.

5 Now we'll use *Flash Amp* to produce a list of figures that can be used in *Flash* ActionScript. Open *Flash Amp*. Click on the question mark to select the AIFF file and hit the Continue arrow. Click it again to confirm the need for an amplitude list.

6 In step 3, tick 'cue point name list'. This will ensure the processing produces a second list of figures that run in parallel to the amplitude list, with the cue points in there indicating when the code should react. Click Continue again.

7 The 'frames per second' indicates how many volume values will be returned per second of audio, matching the fps of your *Flash* file. A higher number will return more values for use with a faster animation, but you must be confident that the file will cope with this speed so go for 15.

8 Step 5 sets the value scale for the readings made from your audio file. Set it to ten. This means the list will show a ten for the loudest sounds and 0 for the quietest. This also means you only need to produce 10 frames of animation from 'closed' to 'open' mouth.



9 Finally, tick yes for 'smoothing' and in the last stage click the text file icon to process the audio. It is saved into a text file which will shortly be used in *Flash MX*. Feel free to open it and take a look at your conversation, now represented as an amplitude list.

Part 2: Making graphics and animation in Flash

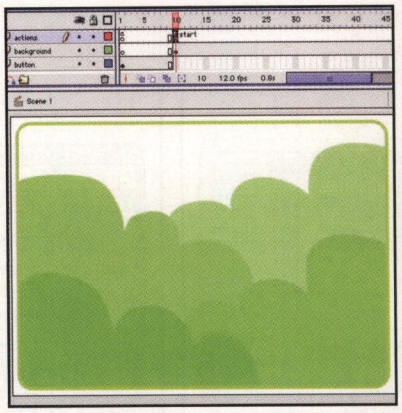
Create your talking characters and synchronise them with your recorded speech…

Keep it clean

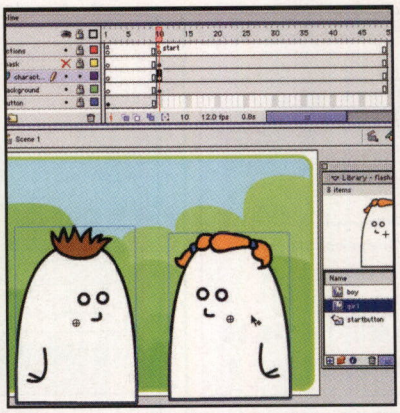
When you start working with background layers, characters, buttons, sounds and ActionScript, it pays to be well organised. Name your layers as you create them and place work on them accordingly. In creating and naming symbols and instances of symbols, make sure your system is clear and watch out for names like arrow, border, break and bullet which are part of the ActionScript language.

Stream sounds

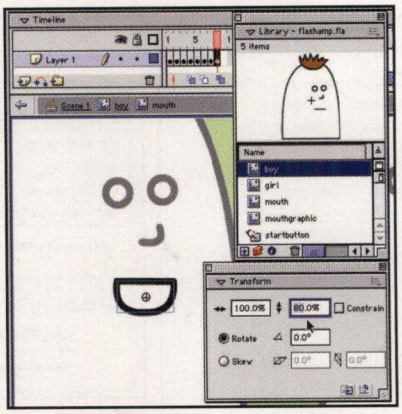
Streaming sound is probably the best Sync setting for this type of work, partly because stream sound stops when the timeline stops. You could load up one long sound in chunks by periodically stopping and then restarting the timeline with user interaction, giving all the frames extra time to load up. Keep in mind that Stream sounds are 'mixed' on Exporting, so any panning across speakers will be lost.



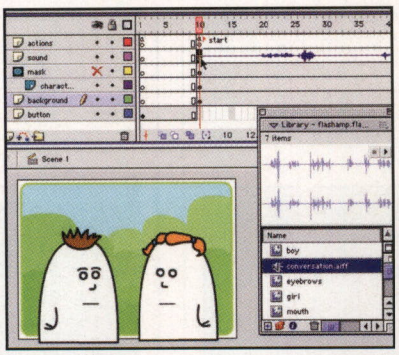
1 Open a new *Flash MX* file. In frame 1 of a Button layer make a button which the user must press to go to the frame "start". Then, in a Background layer, draw a simple backdrop on the "start" frame. Keep it simple so as not to detract from the main characters.



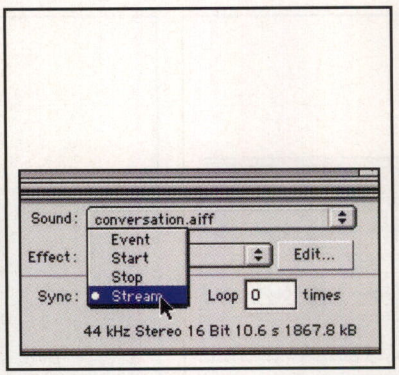
2 Create two movie clip symbols, a 'Boy' and a 'Girl'. Each one should have a face but no mouth or eyebrows. Keep in mind that simple shapes and lines (basically not too many points) will reduce the file size. Place them on the stage facing each other in a masked character layer.



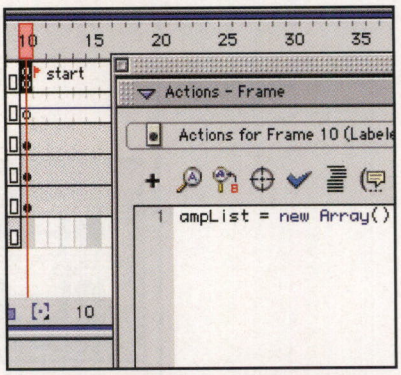
3 In Boy create a movie symbol on a new layer with instance name Mouth. Inside Mouth create a graphic symbol of a fully opened mouth with the height set to 10 per cent in frame 1, 20 per cent in frame 2 and so on, up until all ten frames have been created. Also add a stop() in frame 1.



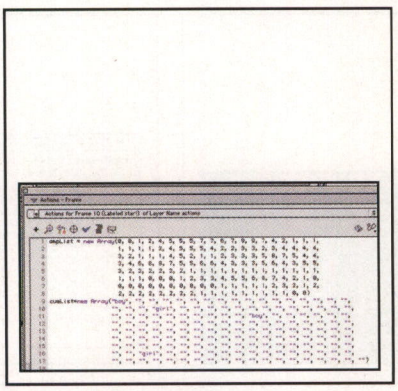
4 Add a two-frame animated symbol to Boy called Eyebrow. The Girl movie symbol should also contain an instance of Mouth – check this in flashamptut_01.fla. Then Import the conversation.aiff to the Library and drop it in a Sound layer at the 'start' frame, stretching all layers to cover the duration of the sound.



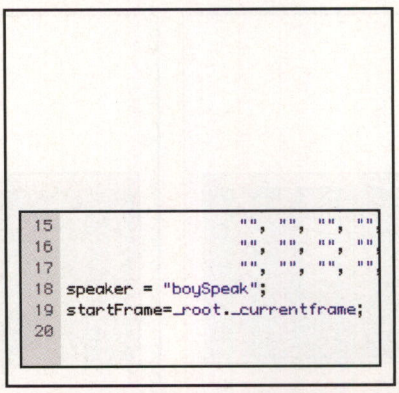
5 Select the frame containing conversation.aiff and set to Sync>Stream in the Properties panel. This means that the Timeline will always be forced into keeping up the same pace as the sound. The sound won't hang up but if *Flash* can't load up frames fast enough, it skips them.



6 In the 'start' frame of the Actions layer, add the following ActionScript which starts up a new Array: ampList. This array (a list of volume values in this case) will be read in sequence as the playhead moves along the Timeline, in turn telling the Mouth clips which frame to show.

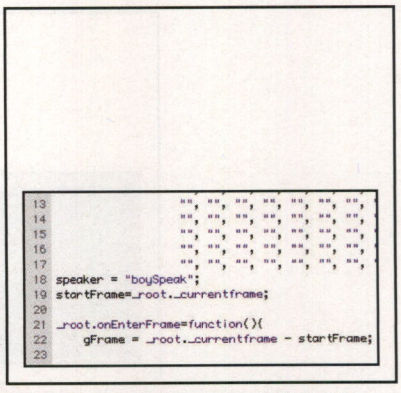


7 Now copy and paste the first list of numbers created in Amplitude.txt and place it within the array's brackets. Create another array called cueList that contains the cue point list from the same txt file, so that your ActionScript looks like this. These lists are now stored in *Flash* for later reference.



```
15
16
17
18 speaker = "boySpeak";
19 startFrame=_root._currentframe;
20
```

8 Under the arrays, add the following two initialising lines. As Speaker will tell the rest of the script which of the two Mouth movies to play, it needs to start on "boySpeak". The constant startFrame is set because the animated sequence of this fla actually starts on frame 10.



```
13
14
15
16
17
18 speaker = "boySpeak";
19 startFrame=_root._currentframe;
20
21 _root.onEnterFrame=function(){
22     gFrame = _root._currentframe - startFrame;
23
```

9 Start an onenterframe loop which starts by returning the frame number that the animation is on – gFrame. This ActionScript sits in the Actions layer from the beginning to the end of the animation, which means that the loop will also be repeated on every frame.

Bangin' choons

Now you know how *Flash Amp* works, you'll appreciate that there are a myriad of possibilities. How about a ghettoblaster in the background which pulsates along to the music it's playing? Rather than using gAmp to go to a frame, you would alter the _xscale and _yscale properties instead. Or how about a bird in the tree which sings a tune? As the volume changes, the top section of the beak _rotates.

```
ampList = new Array(0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                    2, 1, 0, 0, 0, 0, 0, 0, 0, 0,
                    2, 1, 0, 0, 0, 0, 0, 0, 0, 0,
                    4, 5, 5, 3, 2, 1, 3, 5, 4, 3, 2,
                    1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                    1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
                    0, 0, 0, 0, 0, 0, 0, 1, 3, 4, 5,
                    2, 1, 0, 0, 0)
cueList=new Array("",  "",  "",  "",  "",  "",  "",  "boy",
                  "",  "",  "",  "",  "",  "",  "",  "",
                  "",  "",  "",  "",  "",  "",  "",  "",
                  "",  "",  "",  "",  "",  "",  "",  "",
                  "",  "",  "",  "",  "",  "",  "",  "",
                  "",  "",  "",  "",  "",  "",  "",  "",
                  "",  "",  "",  "",  "",  "",  "",  "",
                  "",  "",  "",  "",  "",  "",  "",  "",
                  "",  "",  "",  "",  "",  "",  "",  "",
                  "",  "",  "",  "",  "",  "",  "",  "",
speaker = "boySpeak";
startFrame=_root._currentframe;

_root.onEnterFrame=function(){
    gFrame = _root._currentframe - startFrame;
    gAmp = ampList[gFrame];
    gCue = cueList[gFrame];
```

10 The indexes (values listed) in the array are referenced with the array's name, followed by the index number required in square brackets, starting at 0. So, in adding Lines 23 and 24 to our loop, on frame 10 of the root gFrame will equal 0, so reading 0 from ampList and "boy" from cueList.

```
speaker = "boySpeak";
startFrame=_root._currentframe;

_root.onEnterFrame=function(){
    gFrame = _root._currentframe - startFrame;
    gAmp = ampList[gFrame];
    gCue = cueList[gFrame];
    if (gCue == "boy") {
        speaker = "boySpeak";
    } else if (gCue == "girl") {
        speaker = "girlSpeak";
    }
```

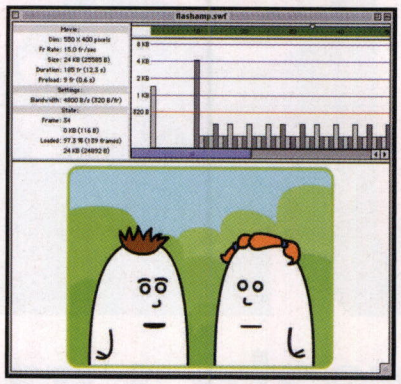
11 The following sets the variable speaker whenever gCue returns Boy or Girl to tell the ActionScript who is speaking (Lines 25-29). As Speaker was initialised as "boySpeak" it will remain so until the animation reaches a value of Girl in the cueList indexes.

```
    } else if (gCue == "girl") {
        speaker = "girlSpeak";
    }
    if (speaker == "boySpeak") {
        boy.mouth.gotoAndStop(gAmp + 1);
    } else if (speaker == "girlSpeak") {
        girl.mouth.gotoAndStop(gAmp + 1);
    }
}
```

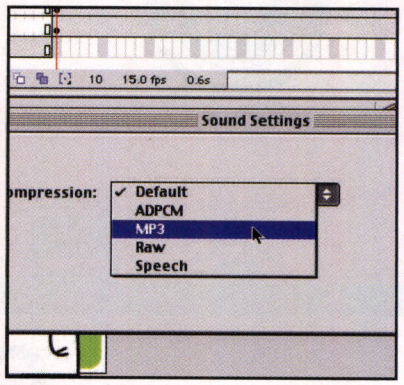
12 Lines 30 and 31 here check Speaker to see whose mouth should be animated, and then the correct frame of either Mouth is shown. Note that a closed mouth should be shown when the ampList returns 0, but as there is no such thing as frame 0, set Mouth to go to gAmp+1.

```
    if (speaker == "boySpeak") {
        boy.mouth.gotoAndStop(gAmp + 1);
        if(gAmp>=7){
            boy.brow.gotoAndStop(2)
        } else {
            boy.brow.gotoAndStop(1)
        }
    } else if (speaker == "girlSpeak") {
        girl.mouth.gotoAndStop(gAmp + 1);
    }
}
```

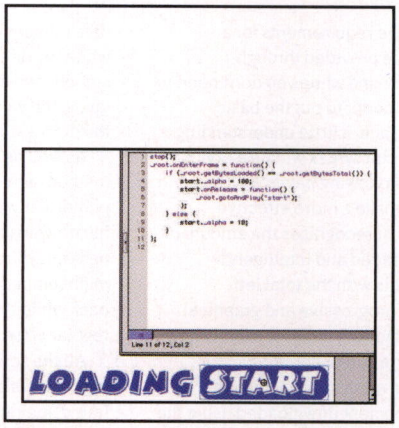
13 Finally, lines 32-36 here show that if Boy is speaking and his voice is raised, the eyebrows follow. When the value drops below 7 again, they return to frame 1 (lowered). Now make sure all your symbol instances are correctly named, check the code against flashamptut_01.fla and test the movie.



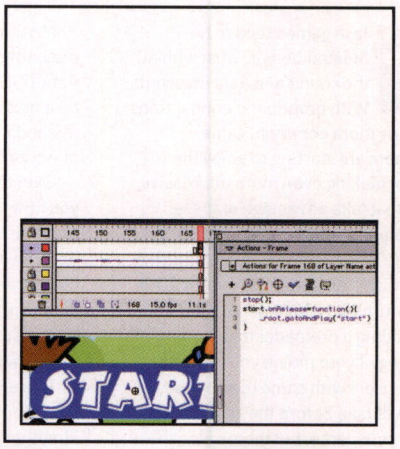
14 As this file contains a moderately large sound file, it may be worth simulating download at different connection speeds from the Debug menu while Testing. Also tick Streaming and Bandwidth Profiler from the View menu. The movie should stream okay if the grey bars fit under the red line.



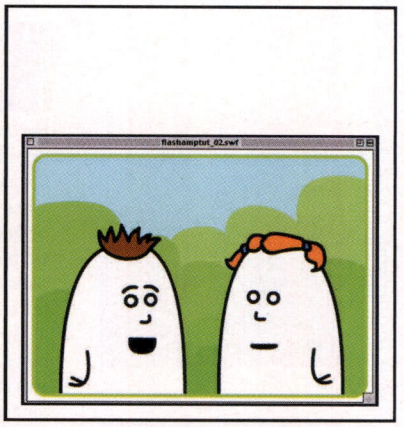
15 If the fla is too large, select the Sound file in the Library and pick Export Settings from the drop-down menu. From here, choose to compress the exporting sound as an MP3, but don't worry – you can always change this back to the original version at any time.



16 If size is still a problem, attach a preloader to frame 1 of the root timeline. As it currently stands, the option to press a button at the start will already stall the viewer enough to let the sound start buffering, but you can find a preloader version in flashamptut_02.fla.



17 As a finishing touch, make sure that all layers stretch right up to end of the sound file (you will see it in the Timeline) and add a stop() action on the Actions layer. You might also want to add a 'replay' button here.



18 It might seem like we've covered a lot of ground but imagine how long it would have taken to achieve the same result without *Flash Amp*! If you look at the final files you'll see how small they are, relying largely on ActionScript and being ideal for slotting into games. ca•s

Illustration: Thomas Brodahl **[w]** www.surfstation.lu **[w]** www.xtrapop.com

Create a games preloader

Preloaders are essential to give some indication of how much of your file is left to come. Here's how you create them...

Flash games tend to be bloated beasts, what with all the sound and data involved. With broadband connections becoming more common, game developers are starting to enjoy the luxury of making even more impressive games that take advantage of these bigger pipes. However, as the majority of UK home Web users are still squeezing data through 56k modems, even a 200k file can turn into a headache.

By adding a preloader to the beginning of your movie you not only provide users with some idea of how long it will take before they can start playing your game, but there's also the opportunity of embedding information such as game instructions, credits or even a mini game to play while the user waits for the entertainment to start.

Most of the requirements for a preloader are provided through ActionScript, and while you don't need to be a master coder to put the basic method in place, a little understanding of ActionScript syntax will be a bonus.

Over the following pages we'll show you how to make a more effective preloader that recognises the amount of data downloaded and intelligently compares this with the total left, providing a progressive and graphical method of showing the user exactly how much more is still to come.

There is a method of gauging the amount of content downloaded using the Timeline properties of _framesloaded and _totalframes but we're going to use a method that compares the getBytesLoaded() with getBytesTotal(). This is more accurate as individual frame data content is often not evenly spread, while the data equivalent provides a smooth and more progressive load rate, giving the viewer a better idea of when the download will be complete.

As well as the familiar filling bar method, we've added a bit more script to numerically show the percentage of the file downloaded. We don't have space here but you can easily see how you might embed other information to play back while the file is downloading using regular *Flash* tools and methods.

All the script that controls the preloader is actually held in a single frame, making this an efficient method not only for developing the original feature but also for minimising any additional weight onto the final size of the delivered file. After all, we don't want to preload the preloader...

Expertise provided by Chris Schmidt. You can contact Chris at **[e]** chris@track5.co.uk or **[w]** www.track5.co.uk

Files on disc

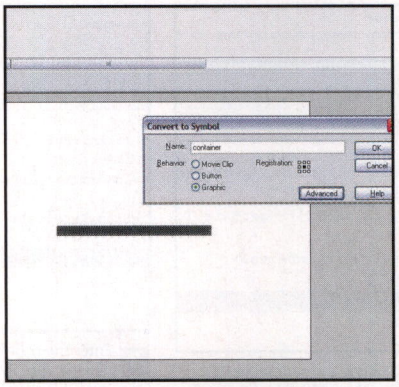
The files needed to support this tutorial can be found in the Tutorial\tech_preload folder on the cover CD.

Part 1: Visual reference

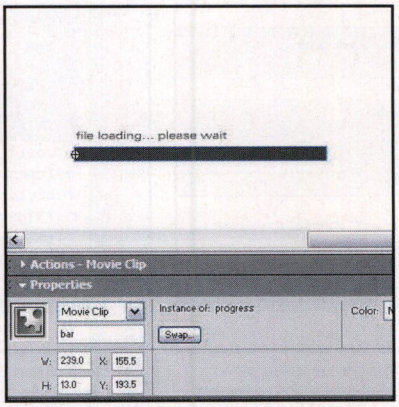
Before you jump into the code you need to get your visual elements established…

Original ideas

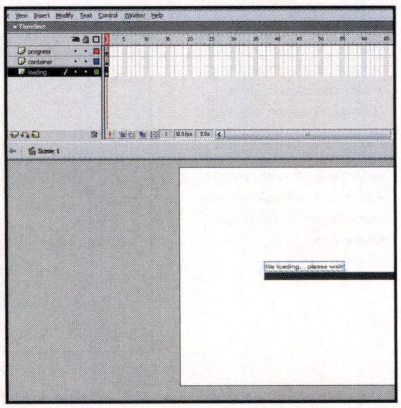
We've used the filling bar idea here as a generic container that is easily recognisable for most users downloading any kind of information. However, *Flash* allows you to use any kind of graphical object as a reference to your file's download status so feel free to play around with other ideas that may be more suitable for your game's theme or design.



1 Draw a long rectangle with the stroke and fill being the same colour. Select the border alone and turn this into a graphic symbol called 'container'. This will become the outline of the percentage bar that defines the extent of the expected download.



2 Next, select the rectangle's fill and convert to a movie clip, making sure the registration point is at its far left edge. Give this an instance name of 'bar' (without the quotes) in the property inspector so we can reference this through the code later on.



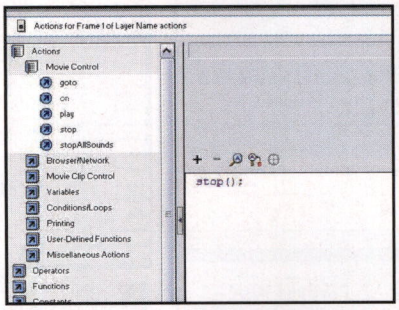
3 With the essential components in place we can now arrange the rest of our layout. We've placed the container and progress symbols on separate layers here and added a little text to make the preloader obvious.

Part 2: Adding the code

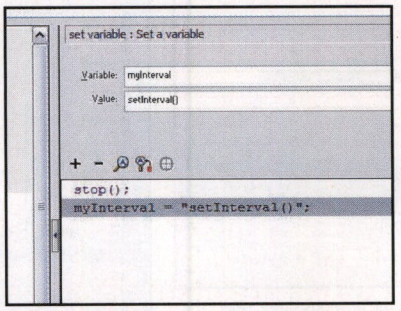
Keeping the code simple will reduce the download and get the preloader operating smoothly…

Something for nothing

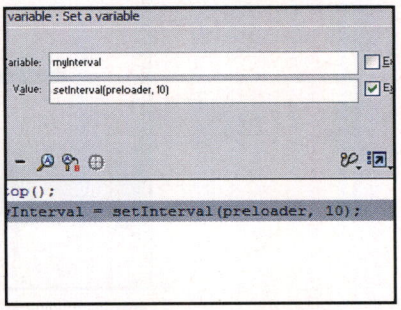
There's obviously no point creating a preloader if there's nothing to anticipate. If you don't have a suitably large .FLA file in existence, create one using heavy graphics or sound files. Similarly, you won't get the full benefit by running the preloader on files stored locally on either your own computer or a local network as the file will almost certainly load instantly. For the most realistic results, load the completed .SWF file onto a remote server.



1 Create a new layer at the top of the stack and call this Actions. Select the first frame and insert a Stop action to restrict the movie from playback until the following preloader code criteria has been met.



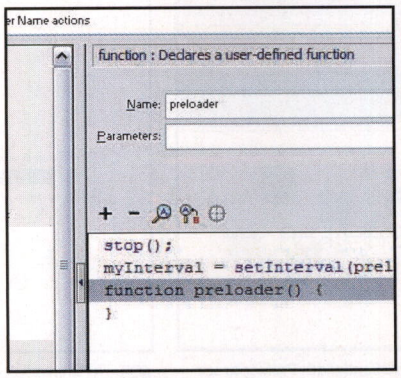
2 Now, through the Actions palette, choose Actions>Variables>Set Variable and enter a name to identify a setInterval action in the Variable field. In the Value field, choose Actions>Miscellaneous Actions>setInterval.



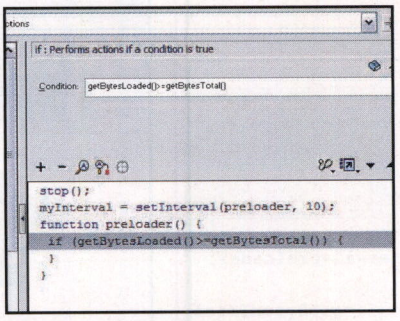
3 Enter the name of a function between the parenthesis of the setInterval action, as shown in the screen above, followed by the interval 10. Now click the expression box. This calls the functions every 10 milliseconds.

Detecting frames

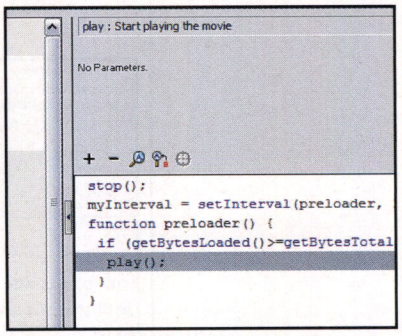
You can easily modify the code here to show the progressive rate that your movie frames are loading in preference to data, as we have here. Simply replace the contents of the Condition field with _framesloaded and _totalframes in place of getBytesLoaded and getBytesTotal respectively, and replace the similar commands in the last statement of the function.



4 To set the actual function, choose Actions>User-Defined Functions>Function and, in the Name Field, enter the name you gave to your function in the setInterval action.



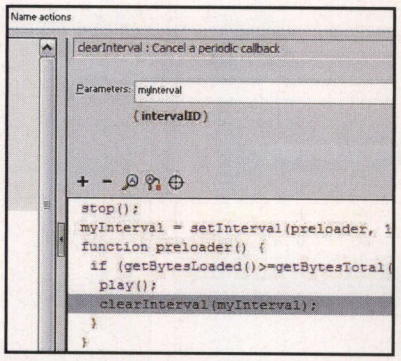
5 Then choose Actions>Conditions/Loops>If and enter the following code into the Condition field: **getBytesLoaded()>=getBytesTotal()** Alternatively, these methods can also be retrieved by choosing Objects>Movie>Movie Clip>Methods if you're not completely confident about entering the code directly.



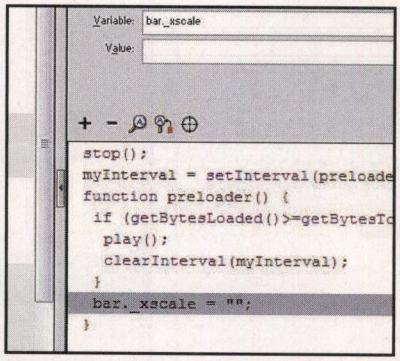
6 The previous step configured the code that compares the amount of data downloaded with the total expected. When this has been met we need to instruct the function on what to do next by adding the Play control via Actions>Movie Control>Play.

Other uses

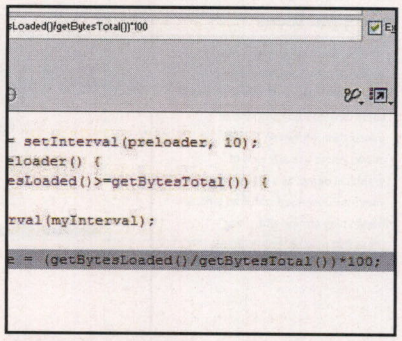
The more obvious use of the kind of preloader we've created here would be for a data intensive *Flash*-driven website. With such simple code already created, you could easily cut and paste the ActionScript before any areas of your site that may require the end user to wait until the download has completed.



1 After all the data has been received we no longer need the setInterval action so we can remove this by choosing Actions>Miscellaneous Actions>clearInterval and entering the name you used to identify the setInterval action earlier.



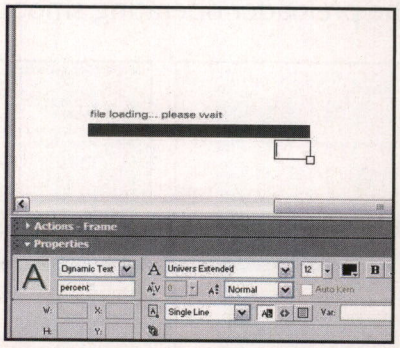
2 The final part is to define the statement that dynamically changes the width of your visual movie clip. Select the closing brace of the If statement and choose Actions>Variables>Set Variables, entering **bar._xscale** as the variable.



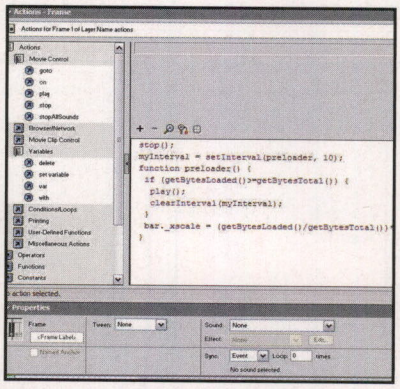
3 Enter the code shown above as the value, making sure to check the expression checkbox. To test the function, create a second frame and import a substantial graphic file to affect the preloader, then upload to a remote server and view to see the preloader in action.

Part 3: Adding a numeric display

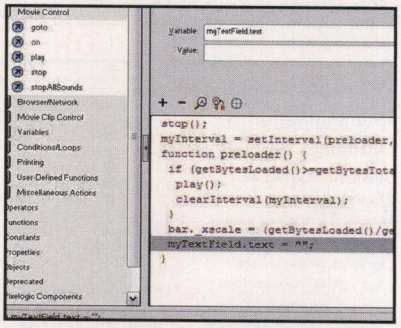
Using a dynamic text field to create a text alternative to complement the graphic approach…



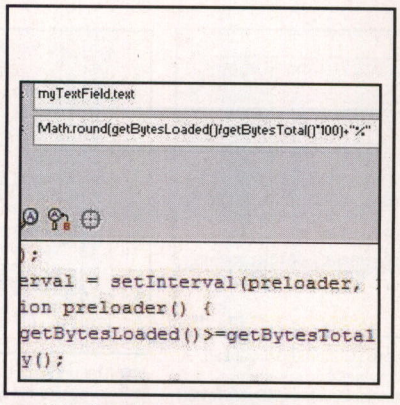
1 Continuing from the previous stage, select the Text tool and drag a text field onto the stage. Give this the name of 'myTextField' in the property inspector so we can target this from the code, as we did with the progress bar previously.



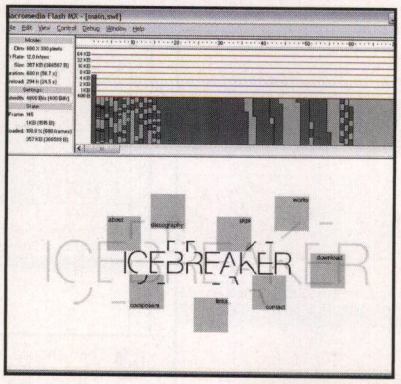
2 Return to the first frame of the Actions layer and expand the Actions palette if necessary to view the code we created previously.



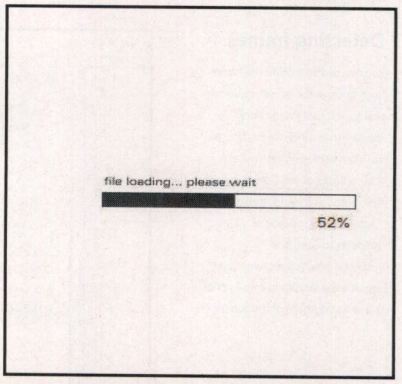
3 Select the last line in the function statement and choose Actions>Variables>Set Variables to create a new line requesting the variable and value attributes. Enter **myTextField.text** as the variable. This refers to the text object which we named in the first step.



4 Enter the value code as shown above, making sure to define this as an expression using the checkbox to the right of the value field. Your code should now be complete and ready to run.



5 You can use *Flash*'s Bandwidth Profiler to see how data is distributed throughout your *Flash* movie when in test mode. First preview the movie through Control>Test Movie and then toggle the profiler on and off via View>Bandwidth Profiler.



6 The final result should provide a preloader that shows a consistent load rate and which provides the end user with some clear indication as to when the complete file will be available using their connection speed. ca-s

NEW! YOUR ESSENTIAL GUIDE

PRACTICAL WEB PAGES

Flash MX

Inside...

EXPERT ADVICE

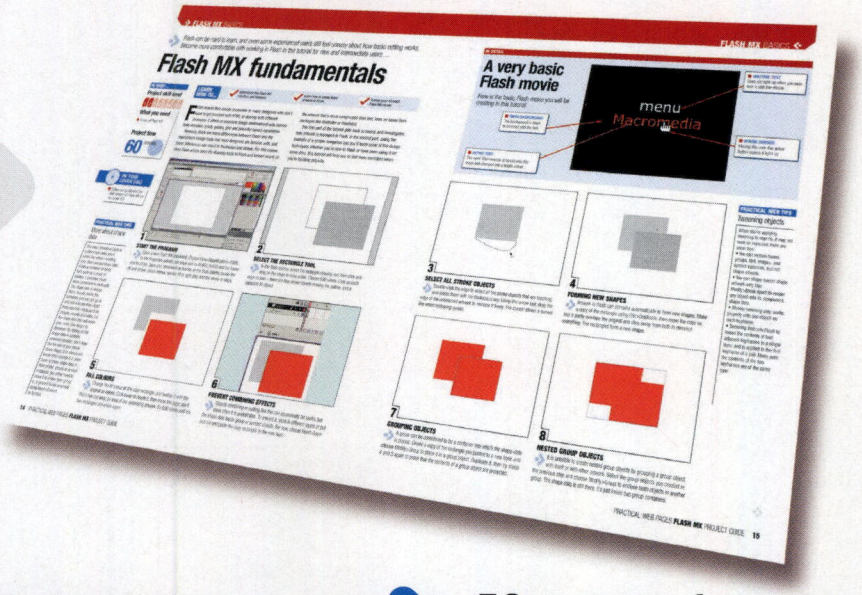
- Advanced tips and tricks from professional Flash MX users

USING FLASH MX

- Find out how to improve your Web site quickly and easily

ESSENTIAL SOFTWARE

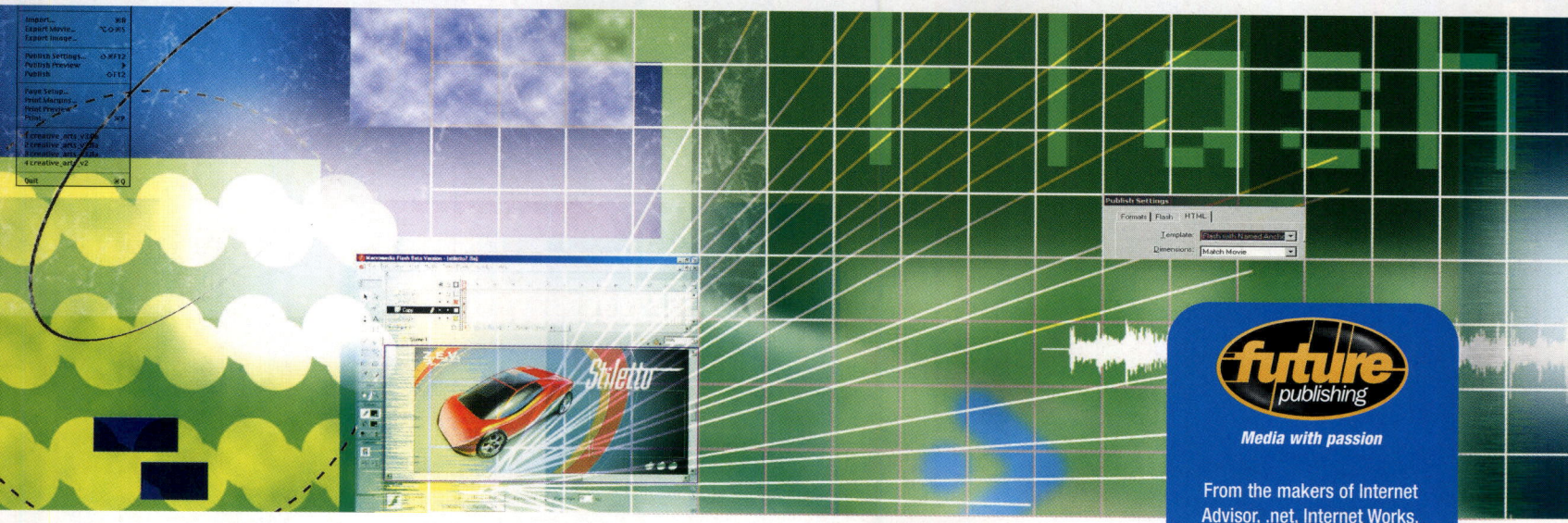
- The latest trial version of Macromedia's flagship web animation software plus a lot more!



20 easy to follow projects for you

50 essential Flash MX tips

50 pages of expert practical advice



future publishing

Media with passion

From the makers of Internet Advisor, .net, Internet Works, Computer Arts and Cre@te online

ON SALE 19TH DECEMBER 2002!

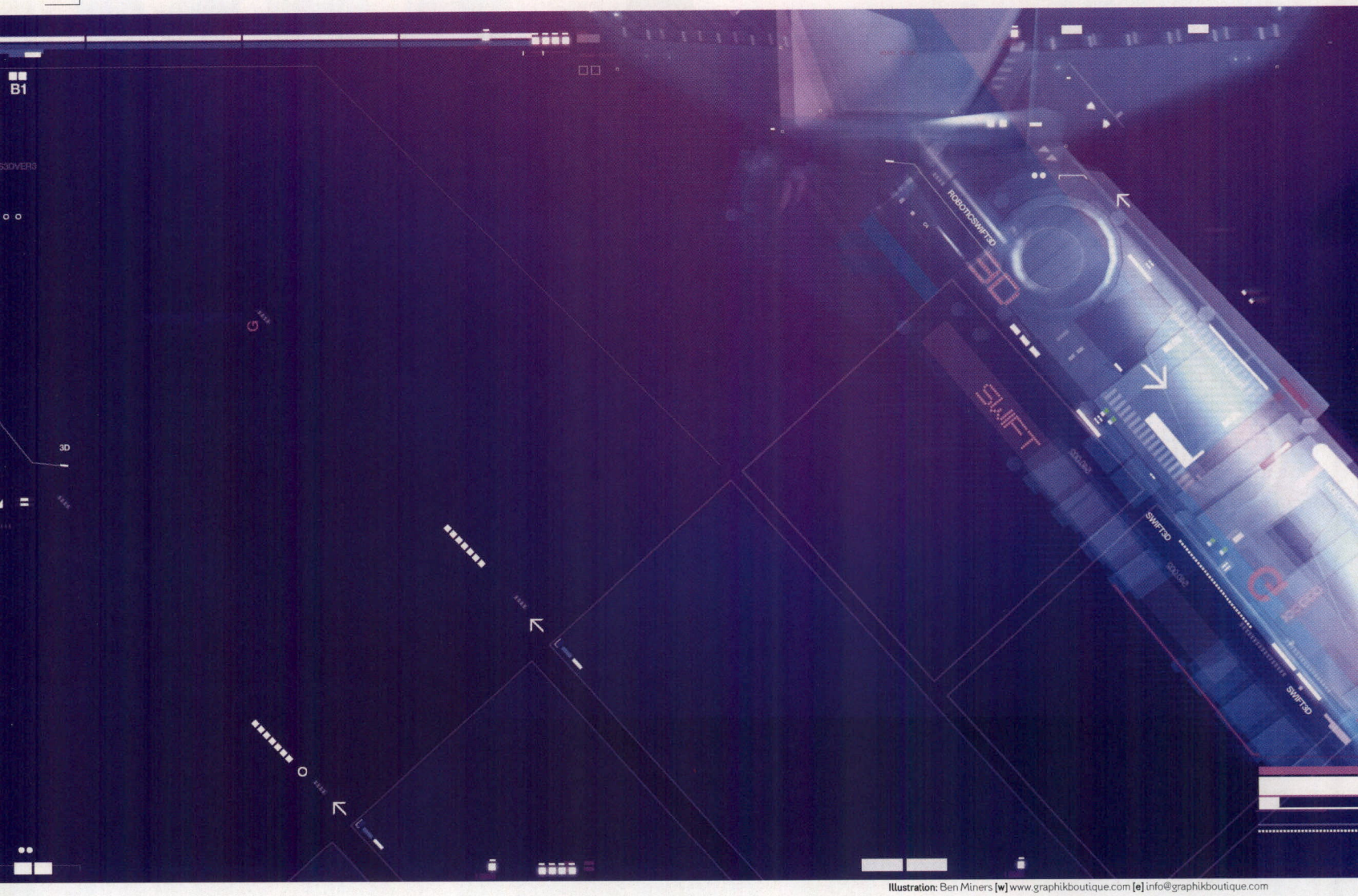
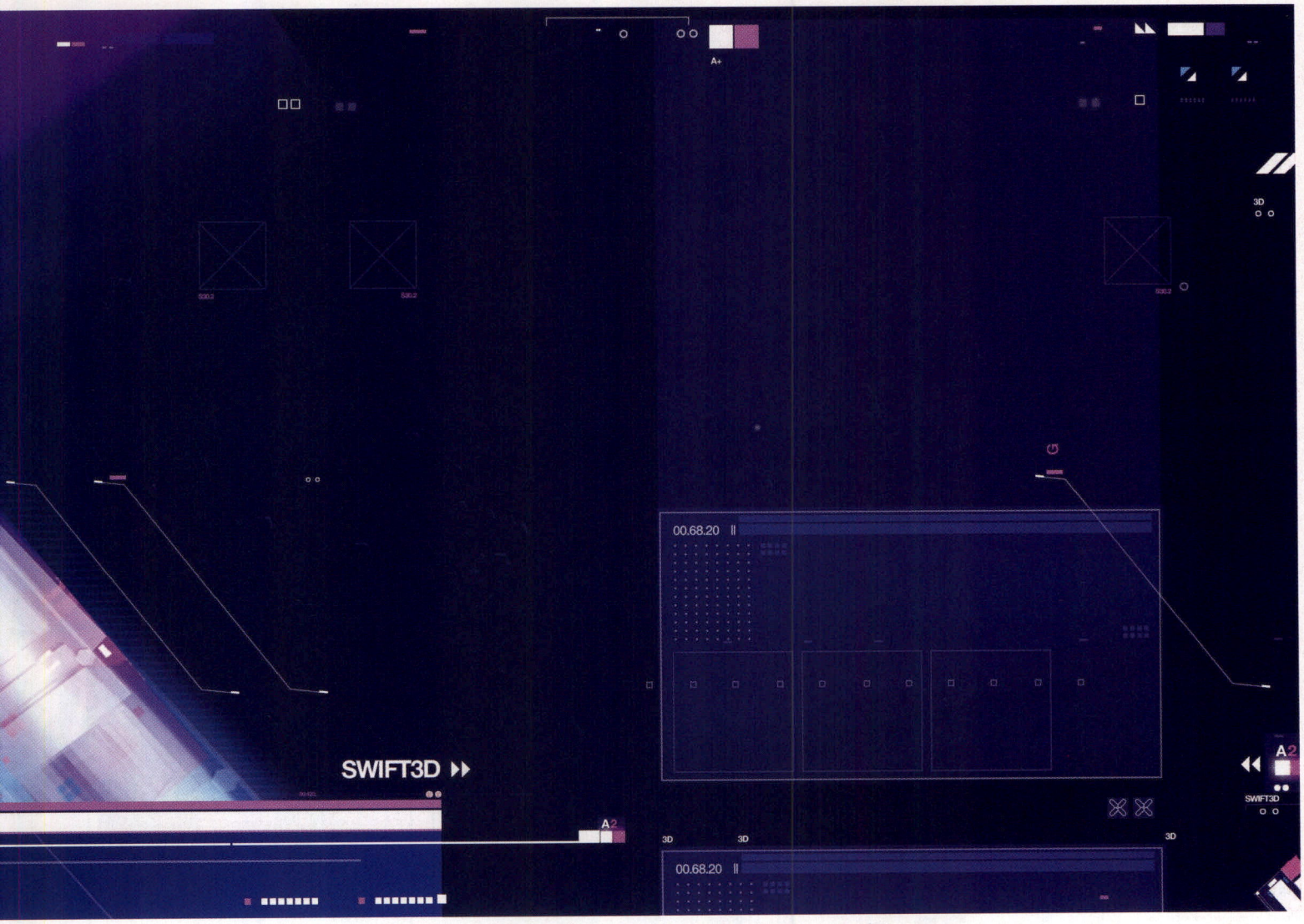


Illustration: Ben Miners **[w]** www.graphikboutique.com **[e]** info@graphikboutique.com



Create a 3D game character

3D and *Flash* are getting it together thanks to the surprisingly powerful Web-friendly *Swift 3D*...

Expertise provided by 3D World regular George Cairns. George is a high-end *Maya* user but is well impressed by *Swift 3D*'s Web-friendly render options. Contact George at **[e]** george@georgecairns.com, **[w]** www.georgecairns.com.

Files on disc
The files needed to support this tutorial can be found in the Tutorial\tech_3D folder on the cover CD.

Now in its third incarnation, *Swift 3D* comes well equipped to let us build a fearsome robotic adversary to liven up a *Flash* game. The cool thing about *Swift 3D* is that you can produce *Flash* 3D animations that take up relatively little file space. The *Swift 3D* animations can also be imported into *Flash* as editable layers. This tutorial will get *Swift 3D* regulars using some of the new tools and features boasted by version 3. The new Hierarchy Editor in particular enables us to animate our droid to display menacing claw-waving action. The novice should embrace this opportunity for a fun introduction to the wonderful world of 3D.

Swift 3D v3 also boasts some powerful new raster rendering options, which enable you to add complex texture maps to your character. However, as our robot is destined for a *Flash* game, we want the finished file size to be as small as possible so we're going to stick to vector rendering, which is something *Swift 3D* excels at.

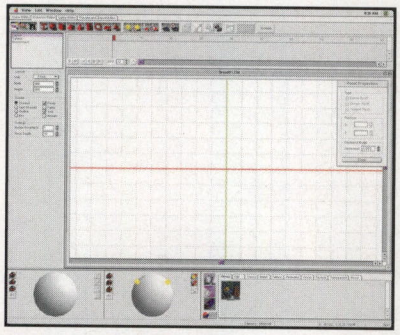
We're going to build a robotic game character because its angular surfaces will suit the vector style renderings favoured by *Flash*. So, get ready for some mechanical mayhem as we unleash a killer robot, *Swift 3D* style!

Part 1: Getting started

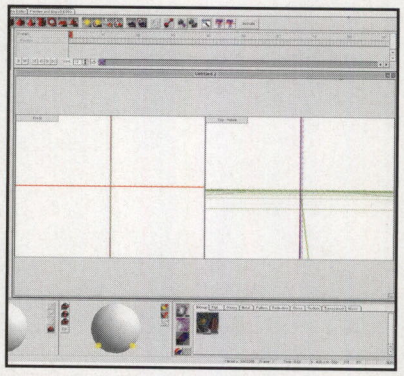
Let's get up to speed with the updated *Swift 3D* interface...

Changing views

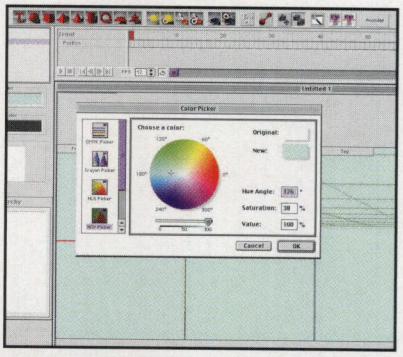
You can toggle between seeing one main window and two smaller ones using the 'Show Secondary camera' icon at the top. The view on the left is the one that goes to full screen. Do keep an eye on your scene from more than one viewpoint from time to time.



1 Start by opening *Swift 3D*. The layout will arrange itself to fill your monitor. It's fully customisable so you can hide various windows depending on the amount of desk space you have. Leave everything on display so you'll become aware of what each window does as you're working through this tutorial.



2 When working in 3D space you may think that your 3D objects are sitting on top of each other, only to later discover that they're miles apart! Using two windows – for example, front and top – will help to avoid unhappy surprises later on.



3 You can change the Background Colour of your scene using the Environment options in the Properties toolbar. However, as we're making a robot to be exported to *Flash*, the Background Colour will be overridden by *Flash*. Only change it if you find a particular colour easier on the eye.

Part 2: Get Modelling

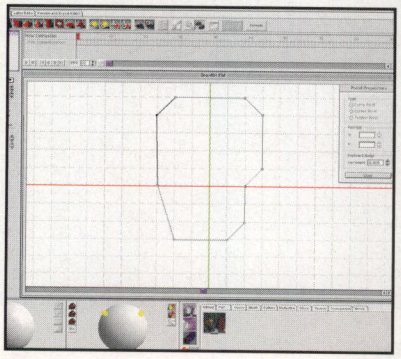
We'll use *Swift 3D v3*'s improved Extrusion and Lathe Tools to create our droid's components...

Adjusting values

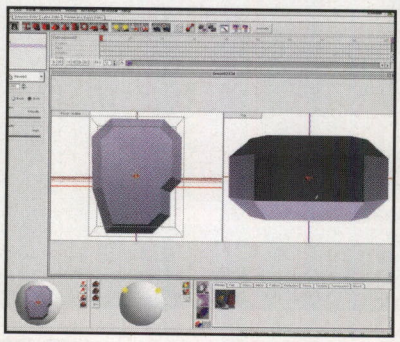
You can change the value of an object's property in several ways. You can either type a new value into the channel box next to the property – Bevel Depth, for example – or you can place the mouse between the up and down arrow next to the channel you want to edit and click-drag to adjust the value interactively.

Ball control

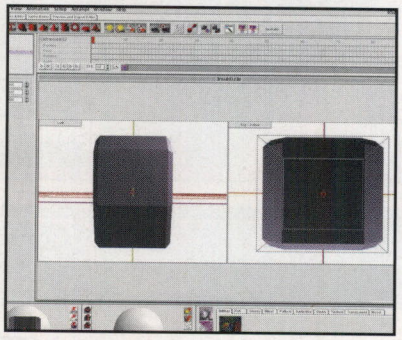
Set the rotation increment of the Rotation Trackball to five degrees. Click the Lock Horizontal icon so the view doesn't rotate out of control. As you click-drag on the Trackball, the Perspective view will rotate and the bevelled shape will disappear out of shot. Now left-click with the mouse to track in the Perspective view until you can see the extruded shape from a new angle.



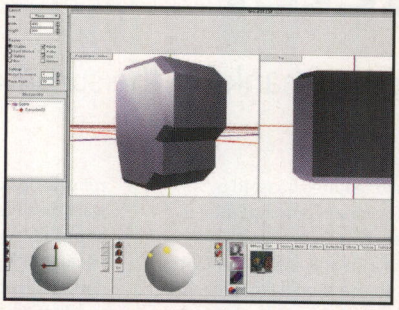
1 Click in the Front view to make it active. Click on the Extrusion Editor Tab at the top of the interface. Draw the profile of the droid's torso by clicking the mouse to place a series of points. Make the last point finish in the same place as the first.



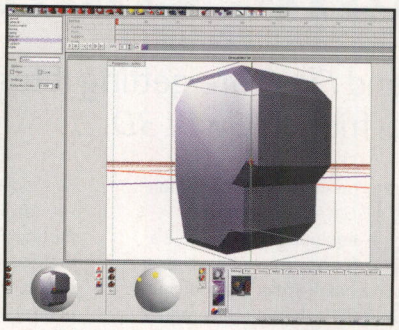
2 Now click on the Scene Editor tab. You'll see that the curve you've drawn has been extruded into a solid 3D shape. Lose the sharp edges of the extrusion. Choose the Bevel option from the Properties toolbar and set the Style to Beveled. Set the Depth of the bevelled edge to 0.02.



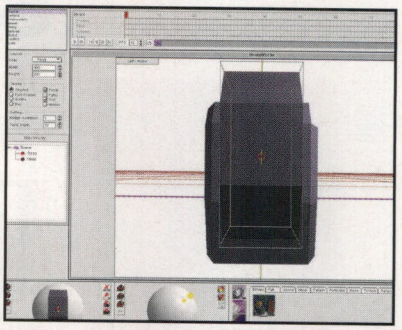
3 Go to the Sizing option and set the Depth of the object to 0.3. View it from different angles by clicking on the tabs in the top left of the view windows. Zoom out by holding right-click and dragging the mouse. Alternatively, control-click if you're using a Mac.



4 Select the Perspective view. Click anywhere in the view window, but not on the object. On the Rotation Trackball at the bottom left of the interface you'll see two arrows appear. Click and drag the ball to rotate the perspective view to see the extruded shape from a more suitable angle. (See 'Ball control'.)



5 Click on the extruded shape. Go to Object in the Properties Toolbar. Label the shape Torso. It's good practice to label your components. As your scene develops you'll get confused trying to differentiate between the many shapes you have if they aren't named sensibly.



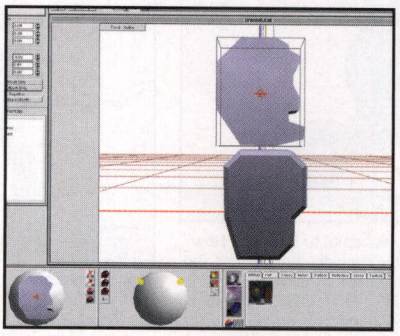
6 Use the Extrusion Editor to draw the outline of a robotic-looking head. Return to the Scene Editor and you'll see that the head shape is overlapping the torso. New shapes are always placed at the centre of 3D space so they need to be moved. Label the new shape Head.

Part 3: Heads up

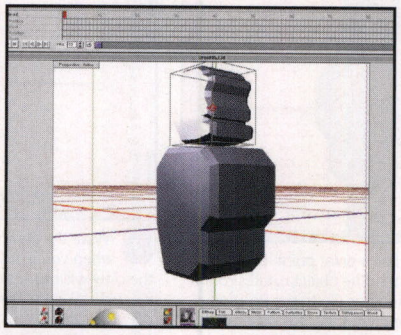
Learn how to adjust the position and size of the objects you create...

Selecting shapes made easy

One way of selecting shapes is to use the new Hierarchy option. If this isn't showing in the Properties toolbar, click on View from the top menu bar and select Hierarchy. In the Hierarchy window, select a shape by clicking on its label.

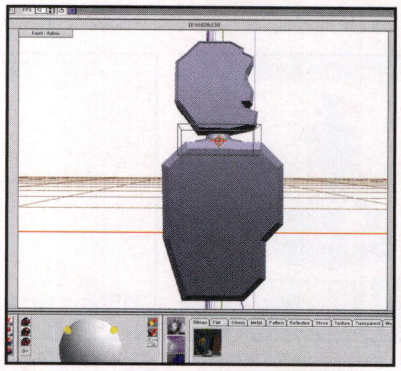


1 Click on the head to select it. Click-drag upwards to position the head above the torso. If the view moves instead of the object, check that the Camera Pan Mode is turned off under Edit. Shift-click to constrain the object's movement in the direction you want to drag it.

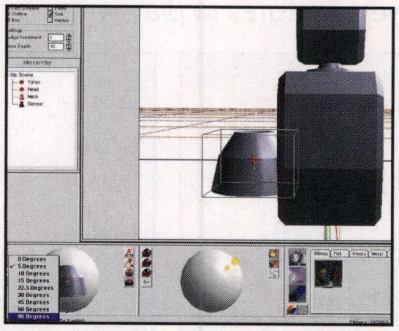


2 Scale down the head using Edit>Scaling Mode. Interactively click-drag to scale the object in the x, y and z-axis. Fine-tune the head's position using the Position option in the Properties toolbar. Place it above the body. Bevel the head just like you did with the torso.

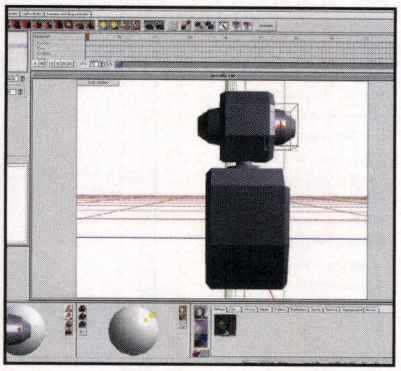
3 To create a neck, go to the Lathe Editor and draw a profile like this. To adjust points once you've drawn them, use the Select tool (the little blue arrow). The Lathe Editor creates a solid shape by rotating the curve you've drawn through 360 degrees around the green y-axis.



4 Go back to the Scene Editor and move the lathed object up in the y-axis so that it rests between the head and torso. Scale it to an appropriate size using Edit>Scale Mode. Label the lathed object Neck in the Object window.



5 Lathe a new shape and label it Sensor. Select the shape in the Left view. You'll see it appear on the Rotation Trackball. Click on the Rotation Increment icon and set it to 90 degrees. Click on the Lock Spin icon. Drag the Trackball and watch the sensor rotate. Scale and position it to fit on the side of the head.



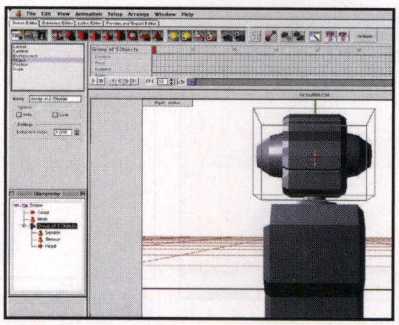
6 Select the sensor shape. Go to Edit>Copy, then Edit>Paste. Use the Rotation Trackball to rotate the duplicated through sensor 180 degrees. Drag it into position on the opposite side of your robot's head.

Part 4: Linking the components

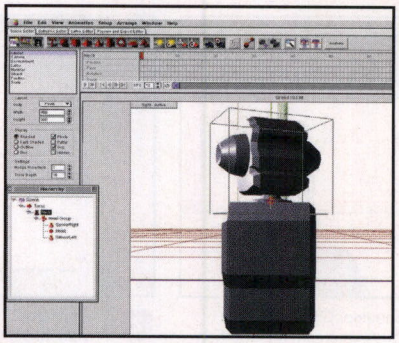
We can give our killer robot moving parts by linking them together in a specific order...

Parent/child relationships

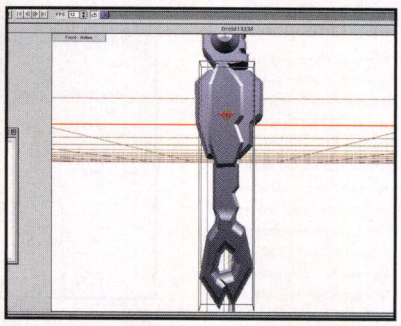
The Hierarchy window is where you link shapes together in a logical way. Drag the Head Group icon onto the Neck icon. The head group becomes part of the neck. When the neck is rotated, the head group rotates with it. The head group has become a child of the neck. By linking up your model in a logical way you can animate its components more easily later. Parent/child relationships are a universal technique that are used in all decent 3D applications.



1 Shift-click to select the head and both sensors. From the top menu choose Arrange>Group. Notice that a Group icon appears in the Hierarchy window containing both the head and sensors. They can now be moved as one by click-dragging them in a view window or rotated using the Trackball. Name the group Head Group.



2 Drag the Neck label onto the Torso label in the Hierarchy window (see 'Parent/child relationships'). Now when you move or rotate the torso, the neck and head Group will move with it. You can still make the neck (the child) rotate independently of the torso (its parent).

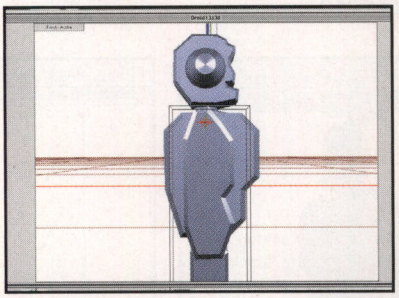


3 Build the upper part of the arm using the Extrusion Editor, then bevel it. Copy the upper arm and scale it down to make a forearm. Add an extruded claw. Position the parts as shown. In the Hierarchy window drag the claw to the forearm and then drag the forearm to the shoulder in order to link them together.

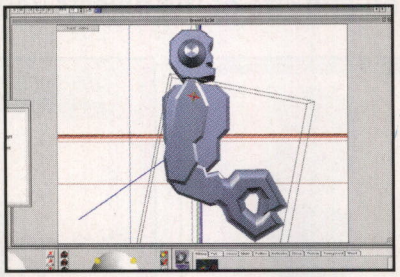
Part 4: Linking the components continued...

Navigating in 3D Space

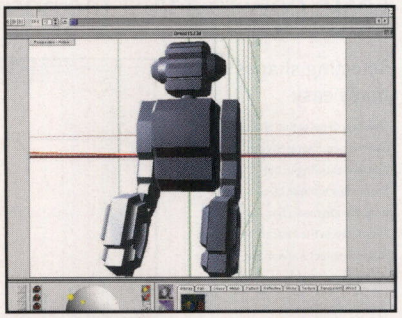
You can zoom in and out of a view. On the PC, right-click and drag down to zoom out of a window and up to zoom in. On a Mac, hold control and drag, unless you have a three-button mouse which will let you navigate in the same way as a PC user.



1 Each component has a pivot point icon that appears at its centre. The object rotates round this point. Select the shoulder. Go to the Position window. Click on Move Pivot Only. Re-position the pivot point as shown. Select the forearm and move its pivot point to where the elbow should be.



2 Move the claw's pivot point to the wrist. Now when you rotate the shoulder, the forearm and the claw will move with it because of the hierarchical links. If you just rotate the forearm then only the claw moves too. If you click on the claw, it rotates on its own without affecting any of the objects further up the chain.



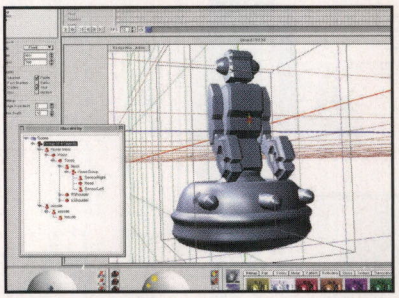
3 Select the Right Arm Group in the Hierarchy window. Go to Edit>Copy from the top menu bar. Then select Edit>Paste. Holding Shift, drag the new Arm Group horizontally until it's positioned on the opposite side of the droid.

Part 5: Texture time

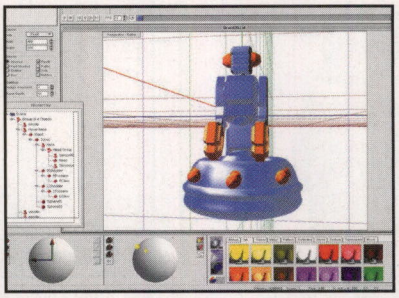
Swift 3D v3 has lots of smart new textures to play with...

Adding texture

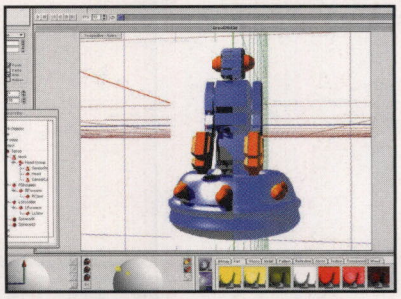
Swift 3D v3 sees dramatic improvements on the texture front. Although we could add Photorealistic textures to our character, we want to keep the file size as small as possible. As it's destined for a *Flash* game we also want it to scale smoothly so we're going to stick with texturing and rendering with Vector-based colours for this project.



1 Finish off the droid by adding an extruded abdomen and a lathed hover base. Top it off by lathing some menacing missiles. Choose Edit>Select All, then Arrange>Group. You'll see a new Group Icon appear in the Hierarchy Editor. You can now select and move the whole robot by clicking on this.



2 Choose a reflective texture from the Material Palette. Drag the material to various parts of your Droid. Drag the texture to the bevelled surfaces too. These can be textured independently, if desired, with other colour styles. As you move the mouse over a texture, make sure a label pops up to say it's a Vector texture.



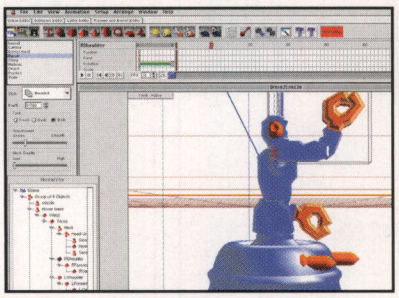
3 Click on the Render Window icon. If you change one component, save rendering time by using the Render Rectangle icon to choose specific areas of the scene to re-render. To make sure the lights are going to glint off that lovely reflective surface, use the Light Trackball to position them high in the scene.

Part 6: Animate and render

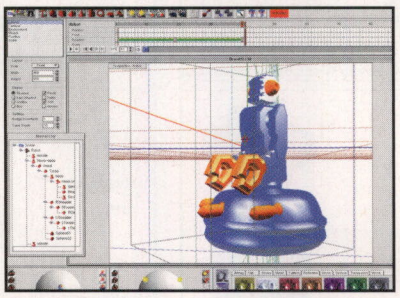
You can make any part of the robot animate over time using keyframe animation...

Keyframing

Keyframe animation is a powerful tool. You use it to set the position of an object at the sequence's start time, then you set a new position for the object at the end of the sequence. The software then calculates all the positions the object needs to go through to get from start position to finish. In *Swift 3D v3* you'll see a green line appear in the Timeline, which indicates the presence of keyframes for an object's particular attribute; for example, Rotation.



1 Select the Right Shoulder Group from the Front view. Click on Animate. In the Rotation Trackball set the Rotation Increment to 90 degrees. Click Lock Spin. In the Timeline go to frame 20. Rotate the arm upwards 90 Degrees. A keyframe is set. Go to frame 40. Rotate the arm back to its starting position. Another key is set. Hit Play to preview the movement.



2 Set keyframes for the robot's whole movement. At frame 40, keyframe each component to be where it was at frame one. Use the Trackball's incremental move option to do this accurately. Now the animation for every component will start and end at the same position, allowing you to loop it in *Flash*. Turn off Animate and position the Perspective view for the final render.

3 Go to the Preview and Export Editor. Set Output options to Vector. Set Target File Type to the new *Swift 3D* Flash Importer (SWFT). Click Generate All Frames. Click Export All Frames. You can now import the animation into *Flash MX*. Experiment with the *Swift 3D* settings. The more colours, detail and reflections you choose, the bigger the finished file will be. Check out our fully scaleable Droid.SWF render on the CD. cd>s

Creative collection^2

The best creative tutorials of 2002, from the world's finest magazines for digital artists

132 pages of expert tutorials on image manipulation, Web design, 3D, effects, illustration and more

- Learn pro skills for Photoshop, Dreamweaver MX, Flash MX, Poser, Illustrator and other leading apps
- Premium-quality printing and cover CD with all the files you need
- An ideal gift for the digital artist in your life

On sale from 28 November 2002 – £7.99 in good newsagents

FROM THE MAKERS OF

COMPUTER arts®

AND

COMPUTER arts special

The in-depth guide for digital creatives

From the makers of PC Plus, PC Format, .net, PC Answers and Internetworks

maxpc

Making the most of your PC

The ultimate PC website. Now with a fresh new look, it's easier to navigate, with speedier downloads and a host of other great features:

- 100s of Reviews covering everything PC
- Expert tutorials to help you get the most from your PC and the internet
- Helpdesk – all your PC questions answered
- Free PC downloads
- Compare prices on PC equipment
- Great offers on books and software

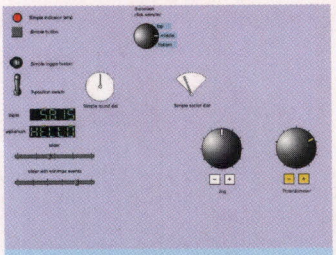
PLUS the latest news, features, competitions and all Future's PC, internet and Creative magazines on the web.

future publishing
Media with passion

Take your PC to the max! **www.maxpc.co.uk**

On the CD: Flash Extensions

Get a headstart in your *Flash* game design and save on download time with our collection of the best ActionScript code examples...

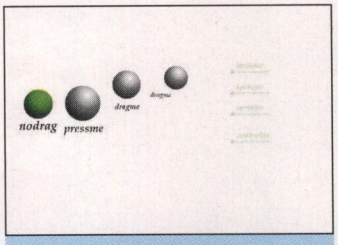


Flash Sim Controls

www.flashsim.com

Those of you interested in creating simulations using *Flash* will find this collection of various simulated electronic and mechanical interface components invaluable. Included in the selection on the cover disc are a joystick, some control knobs and a digital display.

You can find this extension in the EXTENSIONS/FLASHSIM folder.



Active Springer

http://rajaka.narod.ru/english/components/springer/components.html

This is a component for creating physical behaviour links between clips. Simply drag it from the library to the stage, create two clips, name them 'clip1' and 'clip2' and let it run!

You can find this extension in the EXTENSION/ACTIVE SPRINGER folder.

Scorekeeper Extension www.k2w.f2s.com

The Scorekeeper Extension includes tools to create a powerful Web-based high score system for *Flash* games. Note that this extension requires a server that is configured to run Perl/CGI scripts or PHP. This script will not work in *Internet Explorer* on the Macintosh.

You can find this component in the EXTENSIONS/SCOREKEEPER folder.

Car Component

http://robocode.flashflag.com

Simulate the control dynamics of a car. By changing properties such as car weight, acceleration and friction, you can get different handling and performance results.

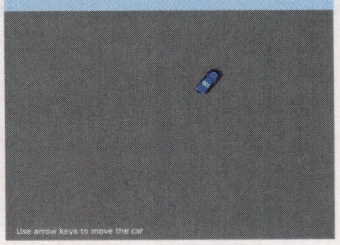
You can find this component in the EXTENSIONS/CAR COMPONENT folder.

Game Control v1.0

www.flashstar.de

Game Control contains four simple game control components: two for car control and two others for controlling a ship. Just drag and drop these components onto a movie clip and set the parameters.

You can find this component in the EXTENSIONS/GAME CONTROL folder.

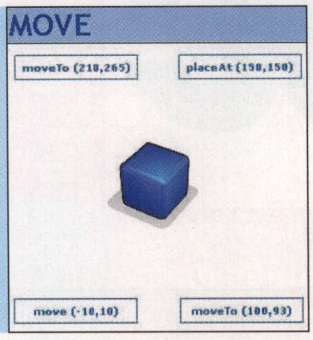


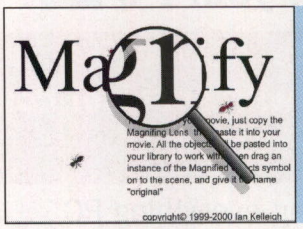
Simple Effects Library

www.flashcomponents.net

The Simple Effects Library contains six different effects components: Zoom, Fade, Rotate, Move, Follow and Framedragger. You can use each of these components to add extra behaviour to your movie clips.

You can find the Simple Effects Library Extension in the EXTENSIONS/EFFECTS LIBRARY folder.



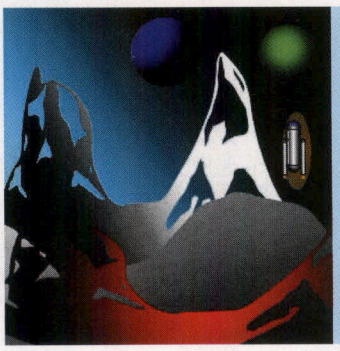


Draggable Magnifier Lens

www.freehandsource.com

This component simulates a magnifying lens to enlarge parts of your movies. The shape and appearance of the lens can be modified.

You can find this extension in the EXTENSIONS/DRAGGABLE LENS folder.



FkeyboardControlBHV

www.tgdd.com

This is a drag and drop behaviour component that can be used to make a spaceship game much like the classic *Thrust*. You can control the orientation of the ship and the power of the engine thrust, enabling you to fly around the environment.

You can find this component in the EXTENSIONS/KEYBOARD folder.

Sephiroth Tetris

www.sephiroth.it

This is simple version of the hugely popular shape-dropping classic *Tetris* puzzle game for use in your movie's preloader.

You can find the Sephiroth Tetris Extension in the EXTENSIONS/TETRIS folder.

Advanced Particle FX

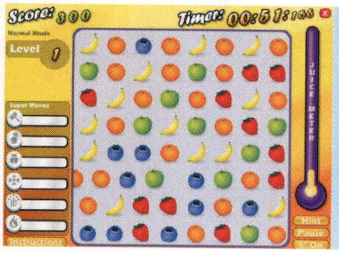
www.szathmari.hu

These smart clips demonstrate how you are able to simulate particle effects such as fire or leaves in the wind, making use of a huge number of repeated objects.

You can find this extension in the EXTENSIONS/PARTICLE FX folder.

Playable Flash Games

We've included ten of our favourite online *Flash* games to both challenge and inspire you to create something that's equally addictive...



Playerthree

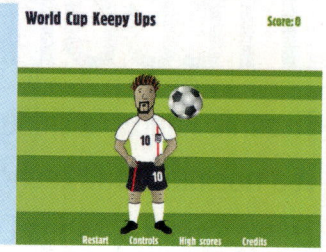
www.playerthree.net

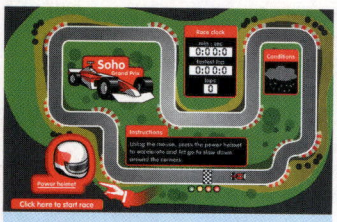
Playerthree creates interactive digital games and toys that are enjoyable, challenging and rewarding. Their clients include The Cartoon Network, Sky Interactive, Panasonic, Swatch, Supernoodles, Technics and Tango.

World Cup Keepy Ups

www.editioninteractive.co.uk

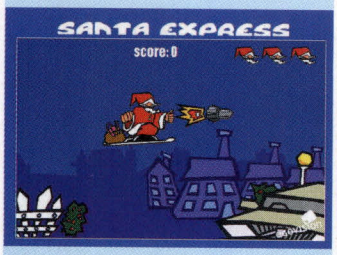
Using Macromedia *Flash*, Edition Interactive specialises in developing database-driven games and applications. In this game you control the football using your keyboard, keeping the ball in the air by heading and volleying.

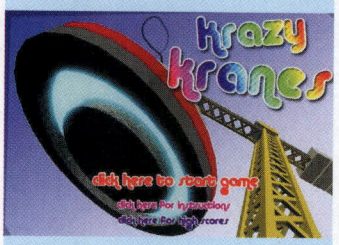




Banja www.banja.com

Banja is a full online community set on the fantasy island of Itland, where you can play games, listen to MP3s, watch movies and chat with the island's other residents. This trailer provides you with an introduction to the unique game world of *Banja*.





Egovision

www.egovision.com

Multimedia agency Egovision has supplied three games for this month's cover disc. *Egoracing* is a Formula 1-themed, top-down racing game, *Egosanta* is a seasonal shoot-'em-up while *Egoscene* is a novel scene-generator application.
Many of Egovision's games are available for rebranding.

//BLOC

www.blocmedia.com

Top London agency //BLOC are responsible for the quirky PlayStation Emotions brand site, as well as the Website for the hugely successful, multi-platform *TimeSplitters 2* game. They're also involved in developing online games for their clients. For this disc they've provided two of their own games: *Stackopolis*, a building block puzzle game, and *Krazy Kranes*, a fun crane simulator game.

Pinpin Lelapin

www.pinpinlelapin.com

Pinpin Lelapin demonstrates how a game can be integrated into an agency site to enable users to navigate through its content. Taking its cues from classic platform games such as *Super Mario Bros*, *Pinpin Lelapin* puts you in control of the title character, exploring a strange environment.

Electrotank

www.electrotank.com

Electrotank creates some of the best *Flash* games on the Web, and we have two demos for you. *Mini Golf Gold* is the highly successful mini golf simulator that pits you against your friends, playing across a variety of terrain and obstacles, while *Fruit Smash* is an addictive puzzle game similar to *Tetris*. The objective is to create lines of fruit and then smash them in order to clear each screen.
Not that both of these game demos are for PC only.

Titoonic

Titoonic knows a thing or two about how to develop *Flash* games. It's not all about technology, though. *"Keep attention to detail and to expression,"* says Creative Director Peter Holm...

Copenhagen's Titoonic is one of the best-known games developers in the interactive design industries. Back in 1989 it was a mere freelance operation, formed by Tomas Landgreen, but in August 2000, Titoonic's current incarnation came about in partnership with Peter Holm. Then in autumn 2001, Henrik Mou, the company's current MD joined, and not long after it became a private limited company.

Titoonic's senior management comes from a post-production background, creating 3D material for broadcast and feature films. But this is a long, pernickety way to spend one's waking hours according to the company's Creative Director Peter Holm.

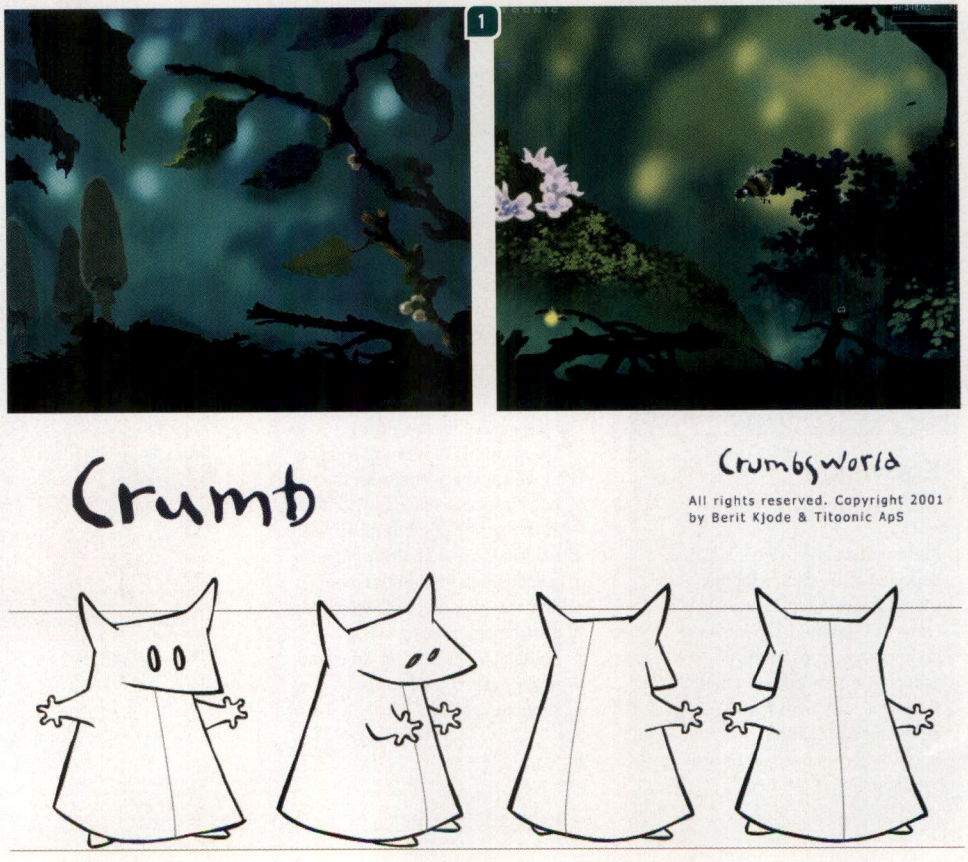
"At some point we lost interest in doing all this hi-res 3D with a lot of textures and mind boggling lights and all this," he explains. "We'd had enough of this long process. We wanted to get into the animation a bit quicker and we could do that by creating stuff for the Web."

Capturing the imagination

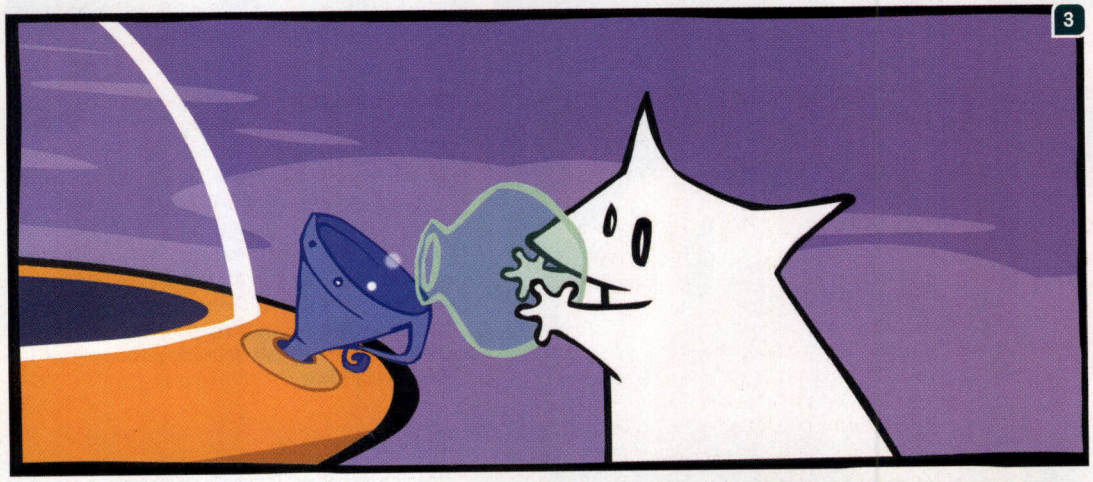
Although the company has developed PC games using technology like C++ and Visual Basic, it is the company's cartoon-style *Flash* games that have captured the imaginations of its domestic and international clients. The list includes Lego, window maker Velux, telecoms company Telia, ISP Chello and confectioner Ferrero. The company employs around 17 people, all of whom have their specialities, although many work across disciplines like design, animation and programming. When projects are underway, they're all divided into teams to make it easier to share information. "We also try to emphasise the importance of asking for help or advice," Holm adds. "We aim at grabbing all the opportunities we get to teach each other and share our knowledge."

There are a few good reasons for Titoonic's extensive use of *Flash* for games development, as opposed to more complex technologies like *Shockwave*. First, *Flash* has a huge installed user base and second, it enables a great deal of creative freedom within a small file. Finally, scripting makes it possible to add complex functions. On the other hand, while *Shockwave* games are more full-featured and include such aspects as 3D interaction, they have a much narrower installed user base. Furthermore, the plug-in is more difficult for many users to get to grips with. "Lately we've had big clients explicitly stating that they did not want *Shockwave* games for these reasons," Holm adds.

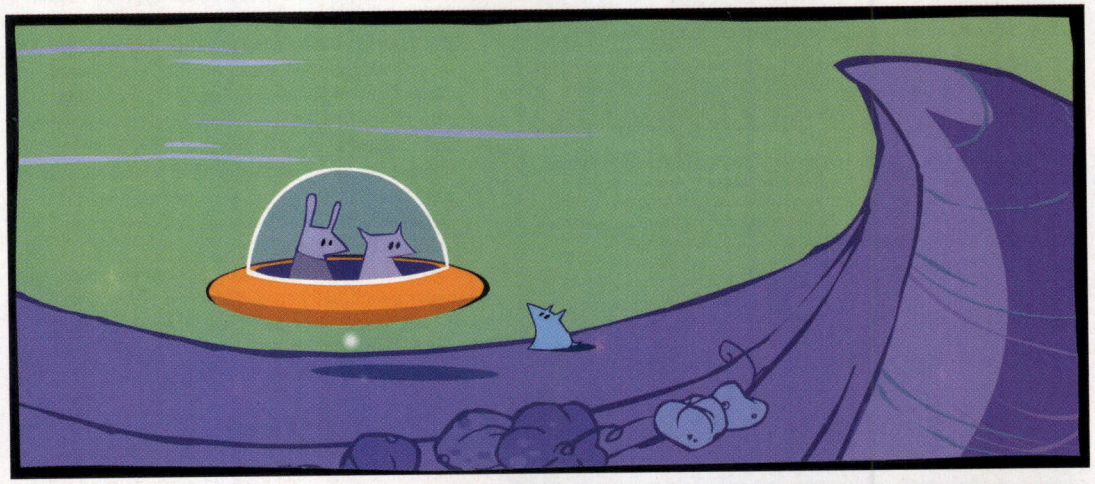
Titoonic's creatives find inspiration in a number of sources. Much of it develops from technical experiments. This, for example is how one of the company's mainstays, its *Snowboard* game, came about. According to Peter Holm, the company was experimenting with ways to create fake 3D. It was highly successful and Titoonic has now sold around 25 customised versions to various companies. While

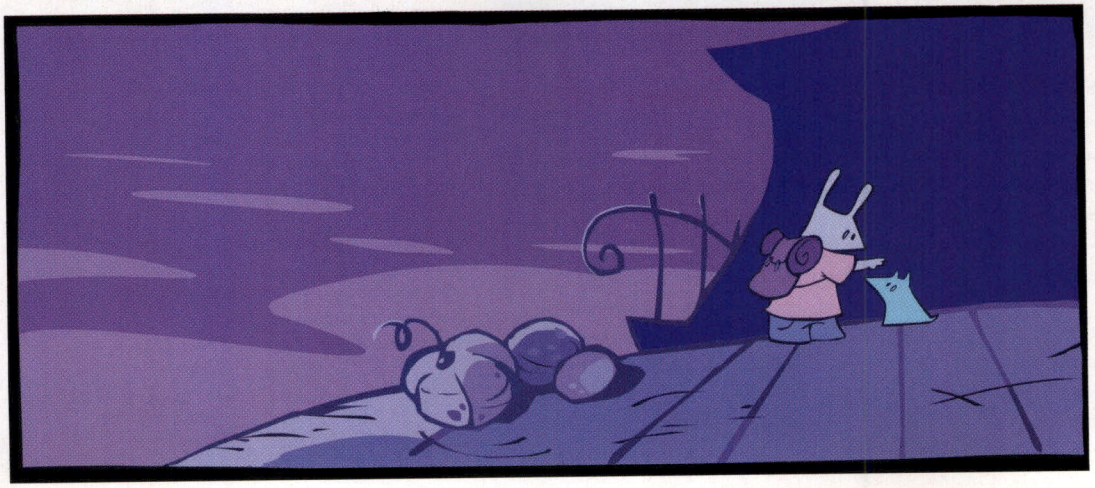


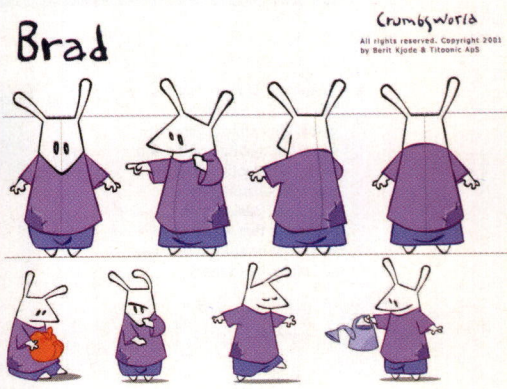
2



3





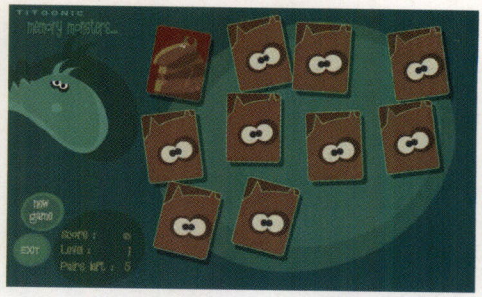


1. Peter Holm is especially pleased with the animation-like programming of the *Spider* game. In it, you take on the character of the eponymous arachnid and your survival depends on the number of bugs you eat. It was one of those Titoonic projects that started out as a technology experiment but evolved into a highly playable game.

2. One of the more well-known versions of the *Snowboard* game is Titoonic's *Powerade* game for Coca Cola, found on the Get Up, Stay Up microsite. It's very easy to spot the similarities between the original game and its custom made, client-led variation.

3. *Crumbs World* is a Titoonic in-house project with its own Website. The company has a view to develop the idea for children's television. In short, it's the story of Brad the pumpkin farmer, his dog, Biskit, and an alien called Crumb, all searching for Crumb's long-lost brother on an Earth-like planet. It was sketched by hand and scanned into *Flash*.

4



4. Another of Peter Holm's favourites is the *Memory* game. It's a simple game of pairs, but the illustration and unique interactivity add an extra dimension. As Holm explains, "Basically, it illustrates that we try to do simple, enjoyable games, but add more life to them."

5

5. Titoonic developed a system for moving a *Flash*-based character around in a 2D/3D space. It dubbed its system the "Isotoonic engine" and it's been used for a number of projects, including one for the Danish National Museum. It's not a multi-player game, but its database integration enables children to interact with virtual objects based on physical materials.

The key is co-operation

Titoonic's clients also include other agencies. The company worked with Framfab on a microsite for Coca Cola...

Titoonic's *Powerade* game typifies the company's work in many ways. It was part of a campaign designed by Framfab for Coca Cola called Get Up, Stay Up. The site of the same name is designed to promote Coca Cola's energy drink *Powerade*, and part of its requirement was a game in *Flash*. That's where Titoonic came in.

Framfab was pretty specific about what it needed. The basic *Snowboard* game needed to be modified to include sports like kite surfing, rollerblading and mountain biking. Much like the original snowboarding game, their central character was built in *3dsmax* and rendered into *Flash*. All animation was done in *3dsmax* as well: "Just with good old manual keyframing, not using *Character Studio* or anything like that," Holm says.

The backgrounds were sketched by hand and then imported into *Flash*, with the outlines traced with red or green bright hairlines. "Sometimes some of our people prefer to use *Illustrator* for this process, but most of the time we do it directly in *Flash*," Holm recalls. It's a fun and simple idea with plenty of longevity.

the snowboard game was developed primarily in *3dsmax* and output to *Flash* using *Swift 3D* or *Vecta 3D* ("I can't remember which one," Holm apologises), old-school techniques are all important at Titoonic.

"Inspiration comes from a lot of very different places," Peter Holm explains. "Sometimes it comes from a dirty pencil-sketch on a napkin that no one was supposed to see but sees anyway. Other times we sit down and think hard for 45 minutes and nothing happens — but the funny thing is, the day after, those who attended the fruitless thinking session will be bursting with ideas."

Tools of the trade

Titoonic's equipment is equally split between digital and analogue. There are lots of PCs: 800MHz PIIIs and 2.4GHz P4s running Windows 2000 and Windows XP, most of which are equipped with GeForce3 graphics cards. There's also *Flash 5*, *Flash MX*, *Photoshop*, *3dsmax*, *Swift 3D*, *Dreamweaver* and *Ultradev*. Music and sound effects either originate from "a cheap microphone and a cardboard tube" or are specially commissioned from Hans Landgreen at BabyCuts (**[w]** www.babycuts.dk) or Jeppe Juul at Cooloop (**[w]** www.cooloop.com).

"Other than that, the list of important hardware is: paper, pencil, ball pen, digital camera, Wacom tablets, a good bicycle and a nice bed," Holm says, smiling.

Gear is one thing, but at Titoonic it's the art of animation, rather than an emphasis on technology, that is most prized.

"I think the trade secret on our part is to keep attention to detail and to expression," says Holm. "Keep an animator's approach. Of course you should worry that the code works properly, but it's important for us that our programmers have a certain insight in how animation works, and what will make animation work. For instance, take the *Spider* game. The spider moves in an 'alive' way, but it's programmed, not animated. That sort of illustrates the idea."

The future holds even more games, using more than *Flash*. Titoonic is finalising its *Java* game engine, for instance, and the company hopes to get into console and Game Boy Advance content. While that sounds daunting, when Peter Holm sums it up, it sounds less like work and more like a joyful walk in the park. "More games, more animation, more stories, more trouble, more fun," he says. Amen. **ca>s**

You can see Titoonic's games at **[w]** www.titoonic.dk, www.getupstayup.com, www.crumbsworld.com. **[t]** 00 45 35 35 93 30. There's a chapter devoted to Titoonic in *Flash 5 Studio*, published by Friends Of Ed.

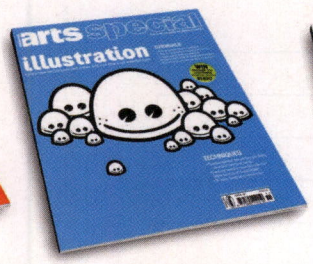
COMPUTER arts special BACK ISSUES

The in-depth guide for digital creatives

Computer Arts Special Get started in digital art – COSB0035
This 100-page issue gives you a complete overview of the digital creative scene, including beginner's guides to: illustrating in *Photoshop 7*, *Painter 7* and *Illustrator 10*; Web design with *Dreamweaver MX* and *Flash MX*; *Flash* animation; creating 3D graphics in *Bryce 5*, *Poser 4* and *D Sculptor*; print and multimedia design using *InDesign 2* and *Director 8.5*. We also show you how to use ActionScript.
CD The full version of *D Sculptor* plus demos including *Flash MX*, *Dreamweaver MX* and *Bryce 5*.

Computer Arts Special MX – COSB0036
An entire issue dedicated to the MX applications *Dreamweaver*, *Flash* and *Fireworks*. Individual tutorials include: developing an e-zine and Web streaming databases in *Dreamweaver MX*; creating motion blur and using sound to control moves in *Flash MX*; site prototyping and animation in *Fireworks MX*. The issue concludes with a big project that uses all three applications to build a virtual radio.
CD The full version of *Flix 1.0* plus demos of *Dreamweaver MX*, *Flash MX*, *Fireworks MX* and *Flix Pro 2.5*.

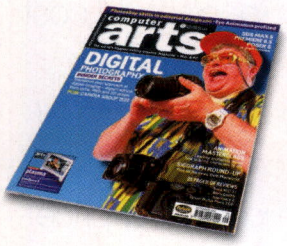
Computer Arts Special 3D Web design – COSB0037
Our complete guide to Web 3D shows you how to: design a Shockwave game, create a virtual multimedia player in *AXEL*, and build a screen saver in *Xara 3D*. We also have tutorials for modelling in *Swift 3D* and *plasma*, and for combining film and 3D footage in *Commotion*. Plus, there's a feature on creating a 3D online store with spinning objects and a round-up of the latest Web 3D books.
CD Full version of *Xara3D 2.1 (PC)* plus *AXELedge*, *Swift 3D v2*, *plasma*, *Commotion* demos and more.



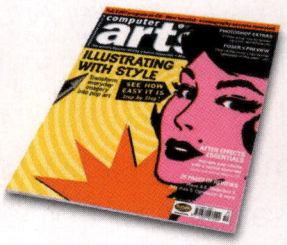
Computer Arts Special Illustration – COSB0038
Learn new illustration techniques and styles by following our practical walkthroughs. Do fashion sketches in *Illustrator*, create pattern designs in *SymmetryWorks*, add perspective in *FreeHand*, combine 2D and 3D in *Photoshop*, produce vector traces in *Streamline*, make textures in *CorelDRAW*, and animate 3D images in *FILTERiT*. You can also find what makes the perfect portfolio, learn 50 easy shortcuts and more.
CD Dual demos of *Streamline 4.0*, *Illustrator 10*, *FreeHand 10*, *FILTERiT 4.1* and *SymmetryWorks*.

Computer Arts Special Typography – COSB0039
Our comprehensive 100-page guide to type shows how you can create fonts in *FontLab*, design a poster and create raised type effects in *Photoshop*, animate type for the Web in *Flash* and produce smooth-moving text for your own film in *After Effects*. There's also our guide to the 30 rules of type, an in-depth look at font management, stencil font design in *Fontographer*, the history and heroes of type and much more.
CD 250 free fonts, full version of *TopStyle 2.51 Pro* (PC), plus demos of *FontLab*, *Fontographer* and more.

COMPUTER arts



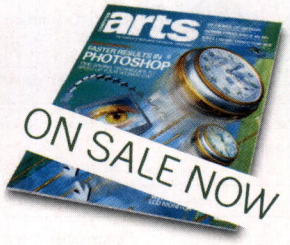
Issue 74 – Code COA74
Digital Photography
CD Demos of the exciting new 3D tools *plasma* and *trueSpace 6*.



Issue 75 – Code COA75
Illustrating with style
CD *VideoClix 1.3* (full), free sounds, *FreeHand 10*, *After Effects* trials.

Issue 76 – Code COA76
Attik, TDR, Mill film speak out
CD *Axel 1.0* (full product), video tutorial on *PS7*, *AXELedge 1.5* trial.

Issue 77 – Code COA77
New series: going freelance in design
CD Fully-working *3ds max* demo including 34 interactive tutorials.



Issue 78 – Code COA78
Faster results in *Photoshop*, setting up in 3D, *Flash* animation, *InDesign*
CD *trueSpace3* full version.

All back issues cost £6. Postage free in UK. Add £1 per issue for European orders. Add £3 per issue for the rest of the world.

CALL THE HOTLINE ON: 0870 444 8455

or TAKE OUT A SUBSCRIPTION – see page 90

☐ **UK** (Direct Debit) £85.00 (SAVE £71)
☐ **UK** (cheque/credit card) £85.00
☐ **Europe** (inc Eire & N America)* £110.00
☐ **Rest of the world*** £130.00

* AIRMAIL

YOUR DETAILS

Title _____ **Initials** _____ **Surname** _____

Address _____

Postcode _____ **Country** _____

Daytime telephone number _____

Email address _____

PAYMENT METHODS

Please choose your method of payment *1, 2, 3 or 4*

1. ☐ **Direct Debit.** **(UK only)** *Complete the form below. Full amount will be deducted in advance and in one lump sum*
2. ☐ **Cheque** *Payable to Computer Arts Specials. Sterling cheques drawn on a UK account*
3. ☐ **Visa** ☐ **MasterCard**
4. ☐ **Switch** **Issue no and valid date** _____

Card no _____ **Expires** _____

Signature _____ **Date** _____

Your subscription will start with the next available issue

UK readers return this coupon (no stamp required) to:
Future Publishing, FREEPOST BS4900, Somerton, Somerset, TA11 6BR.
Overseas readers return this coupon (postage payable) to:
Future Publishing, Cary Court, Somerton, Somerset, TA11 6TB, UK.

☐ **Please tick this box if you would prefer not to receive information on other offers.**

Instruction to your Bank or Building Society to pay Direct Debits.

the future network — Media with passion

DIRECT Debit

Future Publishing Ltd.
Cary Court, Somerton.
Somerset. TA11 6BR

Originator's Identification Number

7	6	8	1	9	5

Please send this form to address shown

1. Name and full postal address of your Bank or Building Society branch

To: The Manager

_____ Bank/Building Society

Address _____

_____ Postcode _____

2. Name(s) of account holder(s)

3. Branch sort code (from the top right hand corner of your cheque) | | | | | | |

4. Bank or Building Society account number | | | | | | | | |

5. Instruction to your Bank or Building Society

Please pay Future Publishing Direct Debits from the account detailed on this Instruction subject to the safeguards assured by the Direct Debit Guarantee. I understand that this instruction may remain with Future Publishing and if so, details will be passed electronically to my bank or building society.

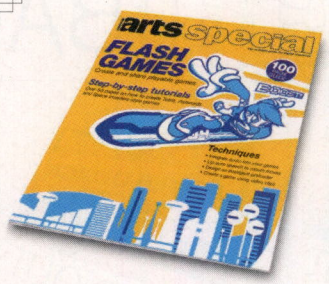
Signature(s) _____ Date _____

Ref No (Office use only) _____

Banks and Building Societies may not accept Direct Debit Instructions for some types of account.

You may photocopy this blank form and then complete it.
You cannot fax or email direct debit forms.

Order code: COSJP40 Offer ends 15.01.03



SAVE £24

Subscribe to *Computer Arts Special* and get 13 issues for the price of 9

Go to page 41 for details

Subscribe to *Computer Arts* and it will set you back £54. And if you subscribe to *Computer Arts Special* it will cost another £54. But, if you subscribe to 13 issues of *Computer Arts* and 13 issues of *Computer Arts Special* here, you'll get them at the discount price of just £85.

That's a massive saving of £71 off the joint cover price of these mags. Call our hotline and guarantee you take advantage of this rate.

All *Computer Arts* subscribers receive a unique limited edition copy with the cover image intact and uncluttered by words.

Did you know all the details you need to complete the direct debit form are on your cheques? Why not fill it in TODAY!

SUBSCRIBE SAVE **£71**

Subscribe to **both** *Computer Arts* and *Computer Arts Special* and get 26 magazines for the price of only 14

That's a whopping 46% discount

That's 12 issues FREE – 6 of *Computer Arts* AND 6 of *Computer Arts Special*

UK 0870 444 8455
Overseas +44 (0) 870 444 8455
www.computerarts.co.uk/subscribe

IMPORTANT: PLEASE QUOTE CODE: COSJP40 **WHEN ORDERING. TELEPHONE LINES OPEN 24 HOURS A DAY.**

G-MAX
SKATEBOARDING

Grey Digital

Winner of two Flash Film Festival awards and achieving cult Web status, you won't be able to stop playing G-MAX Skateboarding. *"The key thing is it has to be addictive,"* its creators explain...

1. As any game should have, *G-MAX* has an intro that captivates and prepares the audience for the task ahead.

2. Remaining faithful to real-life skateboarding, the team at Grey Digital built Flips, Methods, Christs and other popular tricks into *G-MAX*.

1

For those of us a little less 'extreme', who prefer to spend time surfing the Web rather than the streets, *G-MAX Skateboarding* brings all the excitement of Ollies, Nollies, Roastbeefs and Madonnas, without the grazed knees and frequent hospitalisation.

A simple yet completely addictive game from Swedish design company Grey Digital, *G-MAX Skateboarding* was brought to the attention of many when it won in two categories at the Flash Film Festival held in New York in August 2002. Winning Best Game (an honour previously bestowed on Electrotank's awesome *Mini-Golf*) and possibly even more importantly, the People's Choice Award, the game has become a cult phenomenon on the Web, visitors constantly juggling for a better position on the high score table. The premise of the game is simple – score as many points as you can by performing all manner of halfpipe tricks using the Z, X, and Cursor keys. Playing this game, as with the best games in any medium, really is a mixture of enjoyment and sheer frustration.

The company behind *G-MAX Skateboarding* is Grey Digital, a design firm with a client list that includes the likes of Nokia, Procter & Gamble and Electrolux. One of the men behind *G-MAX* itself is Jesper Wahlström, Art Director at GD. We asked him about and how the idea for *G-MAX* was conceived.

"It started as an experiment, like most of our creative ideas do. We wanted to do something for ourselves to promote the fact that we are a serious competitor on the market for online game development. *G-Max* is a non-violent sport game for all ages and we just wanted to make a game that was easy to understand and really fun to play."

In development for around three months, Grey Digital started experimenting with different styles of graphics and animation. Jesper explains that even though the skater is only 35 pixels in height, a lot of effort was put into the animation, making sure movieclip transitions were as smooth as possible.

The game enables you to change the skating environment and even the type and design of skateboard. Considering the board is no more than a few pixels in size, this really shows Grey's attention to detail and the user experience. The in-game graphics are original and stylish, fusing illustrative elements with 3D-like effects. Jesper explains: "Like most of my graphic ideas, I start out with pen and paper; just some raw sketches of the framework and key elements of characters and functionality. Then I transfer all my sketches to *Illustrator* to make all the vector art and the basic elements to work on within *Flash*. The last adjustments and effects I then add on directly in *Flash*, together with the animations."

For all of the 3D animation in *G-MAX*, Grey used Electric Rain's *Swift 3D*, with basic 3D animation being created in the app before being touched-up in *Flash MX*. ActionScripting on the project was completed by Grey Digital's Einar Nilsson, whom Jesper refers to as a "true hero in his field".

G-MAX Skateboarding has certainly found a huge audience on the Web. In fact, after nearly six months of being live, Jesper tells us that over two million people have played it. But what is the successful formula for a *Flash*-based game? Jesper gives his two cents: "The key thing is it has to be addictive, you must get the feeling of just wanting to play the game one more time. The game engine has to be smart, lightweight and ›

2

optimised to make your graphics come alive with the right dynamics and physical motion."

Jesper's inspiration and influences include all kinds of media – TV commercials, movie intros, retro games and new console titles. He tells us that he tries to keep his eyes open to new technical progress and ways in which Grey can improve its graphics. Jesper also gives credit to Nordic colleagues Titoonik, who he says are "true masters of animations and web-based character designs for the Web". Jesper's ideas for new Web-based *Flash* games seem to focus almost completely on the user's experience. The reason behind this approach is clear. "I look back and see what caught my attention in the 80s," he says. "When we didn't have the mega graphic cards and 21-inch screens, we played our games on a Commodore 64 with a small TV as a monitor, but we could stay for hours and play the same game over and over again."

Starting out as an art teacher and freelance illustrator, Jesper became involved in Web design when a company emailed him a proposal for working with something called the Internet. This triggered Jesper's love for interactive design.

"I started out with *Flash* and by the end of 97 I was creating small animations and Web effects" he says. "I'm basically a graphic artist and I had been stuck in *Photoshop* for all my creative Web work. Then *Flash* came along and for the first time a non-technically minded person such as I could combine art with interactivity and functionality on the web."

Jesper and Grey Digital are currently laying down the plans for *G-MAX II*, which Jesper tells us will be more complex and will feature a lot more gameplay, as well as a new high score application. Expect to see it hitting the Web in spring 2003. **ca>s**

Contact Grey Digital on [t] +45 (0) 8 458 0901. You can email Jesper at [e] jesper.wahlstrom@grey.se. Alternatively, check out [w] www.greydigital.se and [w] www.gmaxskateboarding.com.

1

What makes a game playable?

It's all very well creating excellent in-game artwork. Likewise, you can have an entire ActionScript dictionary built into your brain, but these combined do not equal a great *Flash* game. Like any game on any console, *Flash* games for the Web must be playable, because if they're not, and they're not addictive, visitors won't return. *G-MAX Skateboarding* has everything a *Flash* game needs – great graphics, playability and that 'want another go' feel to it. We asked Jesper exactly what he thinks makes an online game playable.

"I believe in instant action. You can't let the player go through all kinds of intros and game descriptions in the beginning and believe he or she has still got their focus on the game when it really starts. On the Internet with online games you don't have more than a few seconds to get the attention you want."

He continues: "You have to convince the player from the beginning that the game is easy and fun, but at the same time a challenge. You have to practise to master it and become a top ten player. The game must give you a competitive feeling and make you want to climb the high score list."

2

3

1. This is an exclusive look at Jesper's artwork for the upcoming *G-MAX II*. Look out for the game hitting the Web in spring 2003.

2. The ability to choose one of seven skateboards is a great example of the amount of detail that Grey Digital put into *G-MAX*.

3. A Christmas greeting from Grey Digital – a *G-MAX* spin-off enabling you to guide a snowboarder down a treacherous slope.

4. Three different halfpipes are on offer at the *G-MAX* skatepark, all with different characteristics.

4

BUY A GIFT THAT LASTS ALL YEAR

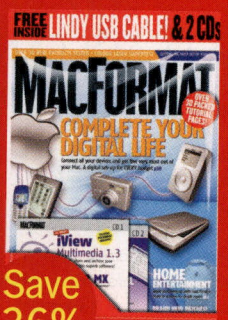
Treat your friends, your family, or even yourself...

CHOOSE FROM THIS FANTASTIC SELECTION OF MAGAZINES:

Save 31%

The best-selling creative magazine for digital artists

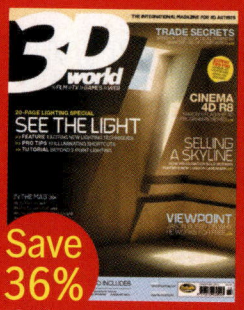
£54.00



Save 36%

Your essential guide to the Mac

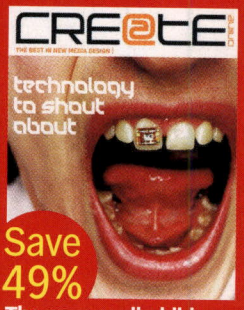
£49.95



Save 36%

The international magazine for 3D artists

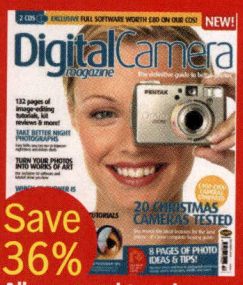
£49.99



Save 49%

The new media bible for designers

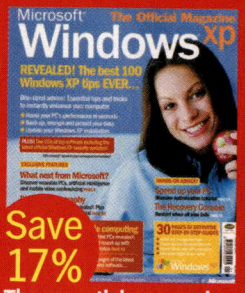
£39.99



Save 36%

All you need to take better digital photos

£49.90



Save 17%

The essential companion for anyone using Windows XP

CD - £64.84

DVD - £69.96

Prices apply to magazines delivered to a UK address only

☎ Order Hotline

0870 444 8470

quoting code 'CREPORTP02' Or subscribe online @ www.futurenet.co.uk/subsCREATE

How to order:

1 Complete the section headed Your Details
2 If you are buying a gift subscription, enter the details in the Gift Subscription section
3 Enter the chosen magazine's title and price
4 Return the coupon, together with your payment to: Future Publishing Ltd, FREEPOST BS4900, Somerton, Somerset, TA11 6BR

Your details:

Magazine title: _____

Mr/Mrs/Ms/Miss: _____ Forename: _____

Surname: _____

Address: _____

_____

_____ Postcode: _____

Home tel: _____

Email: _____

Date of birth: D D M M Y Y

To give a gift subscription:

Mr/Mrs/Ms/Miss: _____ Forename: _____

Surname: _____

Address: _____

_____

_____

Postcode: _____

Two easy ways to pay:

1. I enclose a cheque/postal order made payable to Future Publishing Ltd for the amount of £ _____

2. Please debit my ☐ Visa ☐ Mastercard ☐ Switch

Card No: Switch only

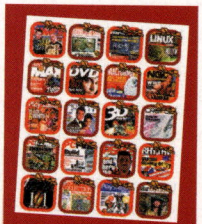
☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐

Expiry date: ____/____ Card issue no ☐☐

Signature: _____ Date: _____

[I am over 18]

Order Total: £ _____



YOUR GIFT CARD

All gift orders will receive a special gift card and envelope for you to add your special message and send on to the recipient. The subscription will start with the next available issue. The final closing date for orders is 28 February 2003. All savings are based on the full newstand price for each magazine. Offer open to new subscribers only. For enquiries and overseas rates, please call +44 (0) 870 444 8470.

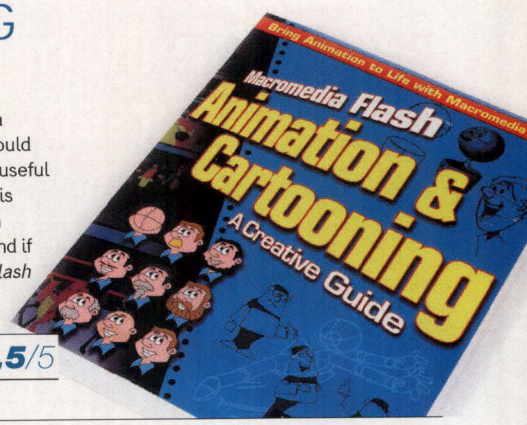
Offer code: CREPORTP02

COMPUTER arts special BOOK REVIEWS

The in-depth guide for digital creatives

MACROMEDIA FLASH ANIMATION & CARTOONING

Price **£29.99** Author **Ibis Fernandez** Publisher **McGraw-Hill** ISBN **0072133236**



If you are looking for a general overview of how *Flash* can be utilised in animation and cartooning then this book is more than adequate. However, anyone who is not an absolute beginner will quickly tire of the basic information presented here. That said, one of the redeeming features of this book comes in chapter 12 where sound and lip-sync are superbly explained for the novice animator.

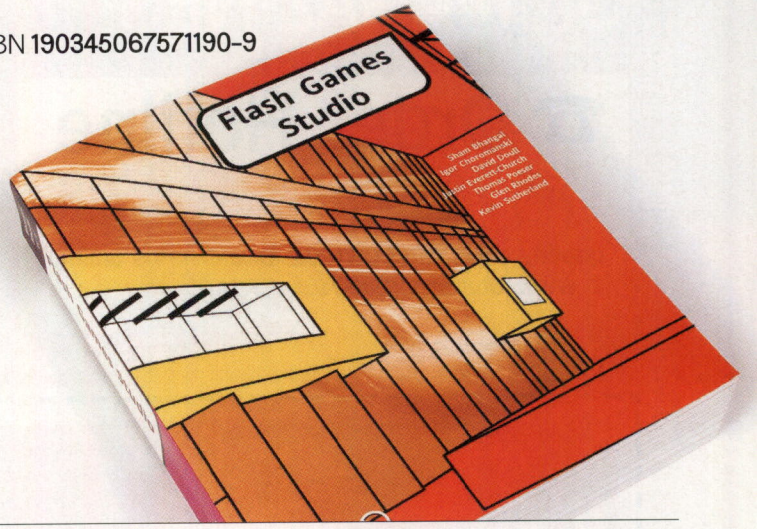
The author begins with a crash course in animation theory, and how these foundations can be enhanced by using *Flash* as a rendering tool. It's not that this book doesn't have some excellent information within its 15 chapters, it's just that it attempts to cover cartooning, animation and *Flash*, but never adequately conveys the all of the information the reader actually needs on any of these.

If you are starting out and need a complete overview of what *Flash* could offer your work, you will find some useful detail here. For everyone else, this is really terrain that has already been covered much better elsewhere. And if you need technical details about *Flash* itself, you won't find that here.

VERDICT **3.5**/5

FLASH GAMES STUDIO

Price **£39.99** Author **Sham Bhangal et al** Publisher **Friends of Ed** ISBN **190345067571190-9**



This book is the third of a series of four that covers *Flash* in detail and so it isn't for beginners — an advanced knowledge of the *Flash* interface and the ActionScript language are necessary.

While not essentially a tutorial-based manual, each chapter does include detailed coding examples, but these offer general instruction on optimising graphics, using *Flash*'s built-in objects and explaining how to handle events.

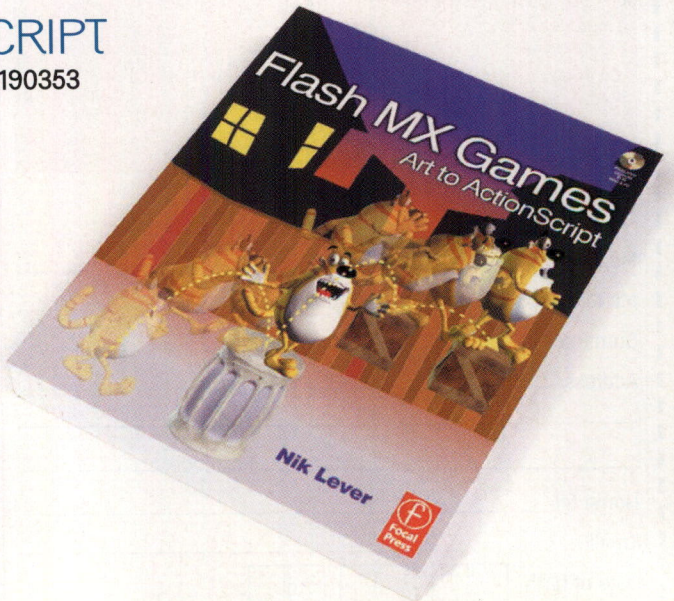
Comprehensive and informative coding examples litter each chapter and are always clearly linked to the effect that is under discussion. You never feel that ActionScript is divorced from the actual game design. This is particularly important in later chapters when 3D and multi-player gaming are considered.

As part of the *Flash* series from Friends of Ed, this book can be read in isolation. However, when placed in context with the other titles in the series, it becomes an indispensable tool. No CD is included so all the coding examples have to be downloaded from the book's Website, but once that's done this book will show you the astounding power of *Flash* when applied to game design.

VERDICT **4**/5

FLASH MX GAMES: ART TO ACTIONSCRIPT

Price **£29.99** Author **Nik Lever** Publisher **Focal Press** ISBN **02405190353**



Superbly written throughout, this is the *Flash MX* book that all aspiring game designers have been waiting for. After a walkthrough of the *Flash MX* interface and some first steps with the ubiquitous bouncing ball, the author then effortlessly explains the intricate detail of the ActionScript language.

Other *Flash MX* books have attempted to explain the ActionScript environment, but this has always been within the context of design for interactive Websites. This book's total focus is on game design. With chapters on quizzes, mazes, board games, platforms and sport sims, your chosen game genre should be adequately covered. Later chapters delve a little deeper into *Flash MX*, with good introductions on integrating it with *C++* and *Director* for some amazing results.

If you have ambitions to design your first *Flash* game, or need to push your existing skills to the next level, this book will show you how. With a CD packed with all the tutorial sample files, as well as example games for you to dissect, this is the perfect companion for all budding game designers.

VERDICT **5**/5

FLASH 5 CARTOONS AND GAMES F/X & DESIGN

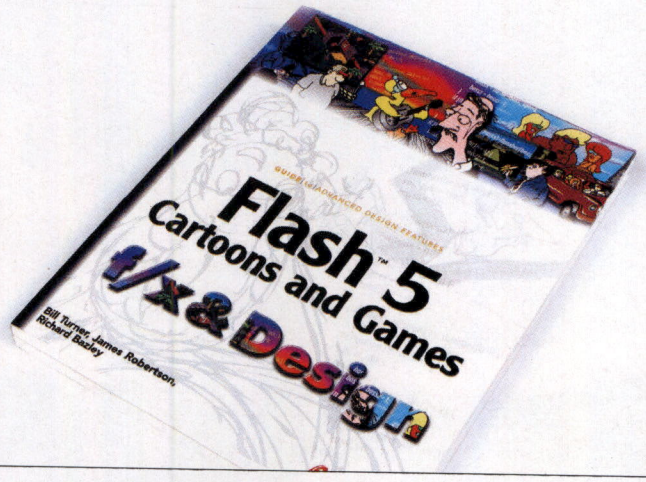
Price **£34.99** Author **Bill Turner, James Robertson & Richard Bazley** Publisher **Coriolis** ISBN **1576109585**

Beginning with an overview of the animation process and the elements that make up the foundations of your work, such as script, storyboard and character design, the second half of this book then switches to game design. Using a wide variety of game types that include a complex isometric projected 3D adventure game, *Flash* as a gaming design tool is explained in detail.

The authors don't stop along the way to explain how a button or feature of *Flash* works so you will need to have quite advanced skills if you are to gain the most from the tutorials included here. This is especially true with the later chapters on game design as ActionScript is presented with little in-depth explanation.

It's rare to find a book that covers not only animation but also game design within the *Flash* environment. With a CD included that's packed with sample files and games to practice with, this is a superb resource for any budding game or animation designer, and the perfect companion for anyone who needs comprehensive advice on either *Flash* animation or game design.

VERDICT **3.5**/5



FLASH MX ACTIONSCRIPT FOR FUN & GAMES

Price **£32.99**
Author **Gary Rosenzweig**
Publisher **Que**
ISBN **0789727994**

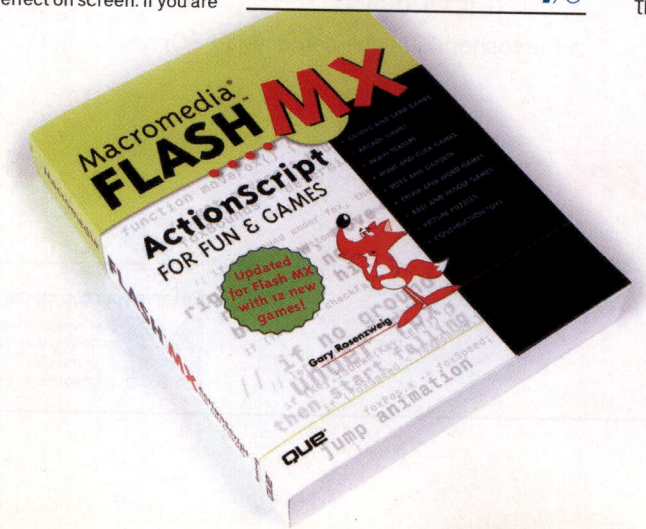
This book has something for everyone. If you are a beginner, the introductory chapters on ActionScript teach you all the basic, essential skills, while later chapters offer more experienced coders the chance to learn some new tricks.

This book has been overhauled to incorporate the *MX* upgrade, as well as including new games in its text. Beginners will need to read chapters one to 15 before attempting to customise any of the coded examples that are included throughout the rest of the book, although more experienced users will be able to jump right in.

The ActionScript code is expertly related to the effect on screen. If you are happy with the input and manipulation of raw code then this book is ideal. While the dense presentation of the ActionScript may at first seem to be impenetrable, it is in fact presented in a logical format that most *Flash* users should be able to cope with.

The basic functions of ActionScript can be learnt from dozens of other books, but this is one of the few that relates this knowledge to game design. Once you have grasped the building-blocks that all games are built on, it's then just up to you to add your imagination and creativity.

VERDICT **4**/5



FLASH MX GAME DESIGN DEMYSTIFIED

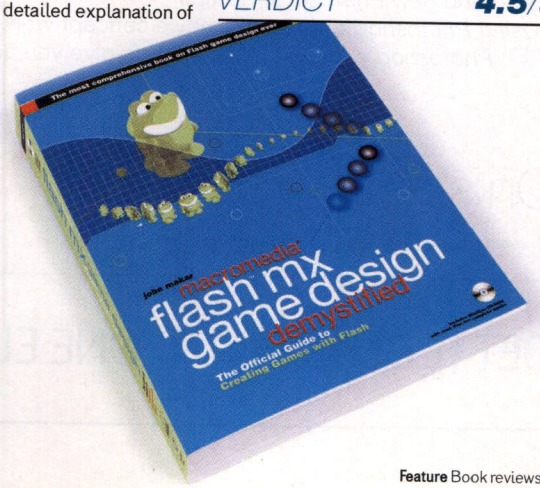
Price **£37.99**
Author **Jobe Makar**
Publisher **Peachpit Press**
ISBN **0201770210**

The first third of this book teaches the essential skills that you will be putting to good use in the later chapters that look closely at actual game design. Basic physics is covered, as is collision detection and tile-based worlds, before concluding with a discussion of artificial intelligence and how you can incorporate these traits into your character design. You even get a refresher course in trigonometry!

Part three then systematically dissects a selection of games, each carefully chosen to illustrate and explore the fundamental gaming rules. Platform games, shoot 'em ups and multi-player games all come under the spotlight. There is plenty of detailed explanation of the ActionScript for each game that allows you to see it in the context of the game itself.

Concluding with excellent supporting appendices, the best of these covers ElectroServer, which is used extensively in multi-player games. Also included is a detailed discussion of the ActionScript object ElectroServerAS. This superbly supports a book that is intensely practical as well as informative. Intermediate and advanced *Flash MX* developers should look no further for a complete, comprehensive and concise overview of *Flash* game design. ca-s

VERDICT **4.5**/5



NEXT ISSUE

ISSUE 41: PHOTOSHOP

In-depth tutorials

- An action-packed movie scene using the Clone tool
- Spray-can effects in graffiti-style images using textures
- Complex fantasy 3D images with layer masks

Features

- 30 brand new *Photoshop 7.0* tips
- Using *Photoshop* to produce movie concept work
- Top *Photoshop* experts and illustrators give you advice

Techniques

- Slicing images and controlling compression
- Customising brushes, actions and shapes
- How to set up a batch process
- Colour management with digital photography – explained!
- How to create graphics for mobile phones

On the disc

- Photoshop 7.0 – full demo! (Mac/PC)

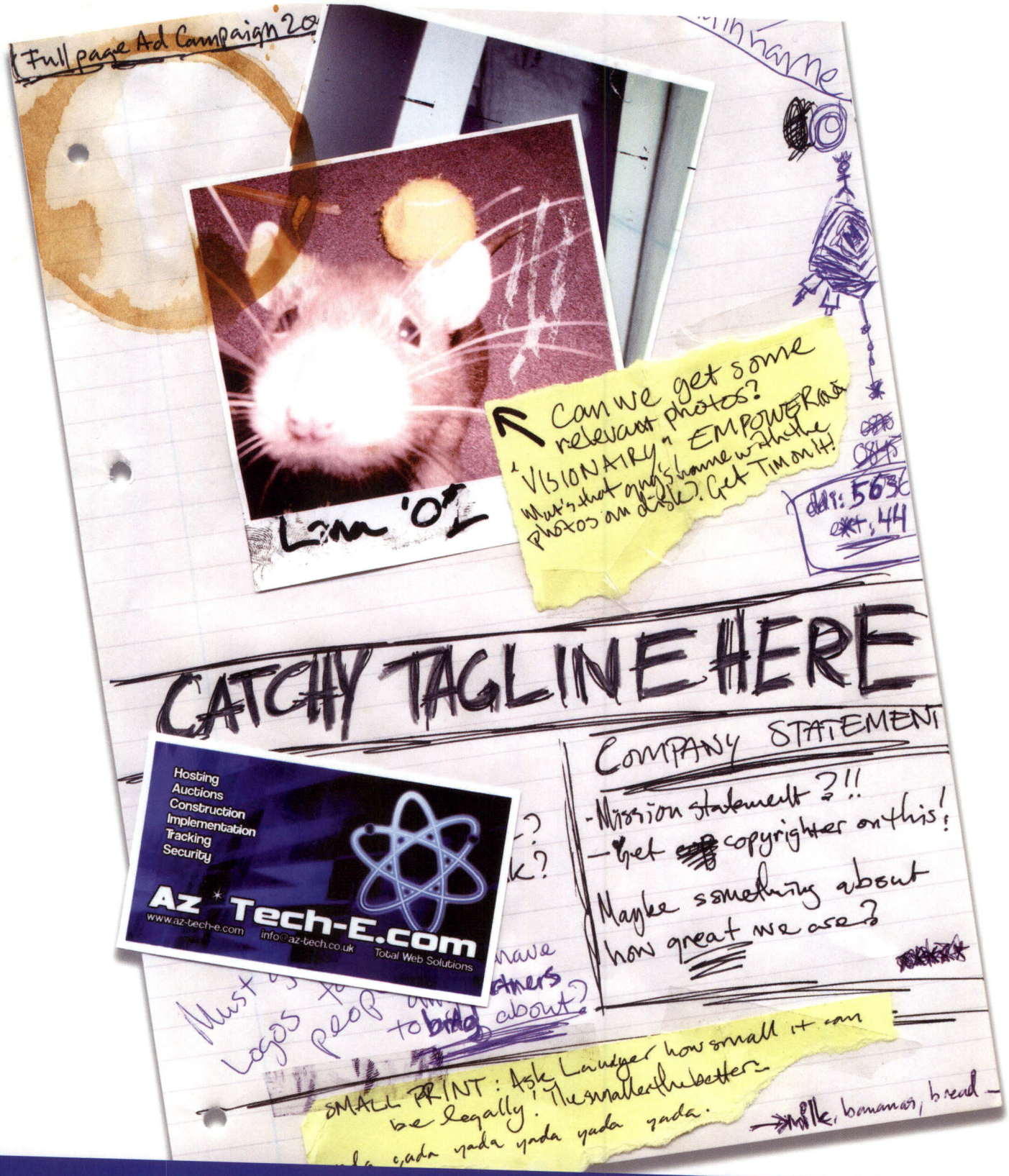
On sale Thursday **16 JANUARY 2003**

Updates at [w] www.computerarts.co.uk

GET IT BEFORE ANYONE ELSE.

Reserve your copy today

Ask your newsagent to reserve or deliver your copy of *Computer Arts Special* to your door every month.



All Service... No Spin

Az*Tech E.com hosting solutions include as standard:

Domain Name Registration / Transfer
Personal Control Panel
Full FTP Access
Unlimited POP3 boxes
Unlimited E-mail forwarding
CGI-bin enabled

For more details and other services :

www.az-tech-e.com

sales@az-tech-e.com

Business	£10 pcm £120 billed yearly
Business Plus	£13.25 pcm £159 billed yearly
Business Silver	£20.75 pcm £249 billed yearly
Business Gold	£29.92 pcm £359 billed yearly

Webspace (MB)	50	100	200	300
Live statistics		✓	✓	✓
ODBC support		✓	✓	✓
Mirrored mail server			✓	✓
Extra processing power			5x	10x
Low population server			✓	✓

* This is the small-print section, where we tell you that all prices quoted above are ex-VAT and that VAT at 17.5% will be added to the cost when invoiced. Prices, services & availability can change without notice, but probably won't.

One award doesn't make Canon the leader in digital technology.

But then, we didn't win just one.

EOS D60 European Pro Digital Camera of the year 2002-2003

EOS-1D Best SLR Digital Camera in Europe, 2002-2003

PowerShot S40 Best Digital Consumer Camera in Europe, 2002-2003

PowerShot G2 European Digital Camera of the year 2002-2003

EF 400mm f/4 DO IS USM Best Lens in Europe, 2002-2003

At Canon, we think the combination of optical know-how and state-of-the-art imaging technology has created the finest range of digital cameras available. We are delighted that the experts think so as well.

www.canon.co.uk

you can

Canon